



Universiti Malaya

**FACE RECOGNITION USING
EIGENFACE / PCA
(WXES3182)**

**MUHAMMAD FADZLI
BIN
AHMAD JALALULMAHALLI
(WEK 010 359)**

**Supervisor:
Assoc. Prof. Dr. Roziati Zainuddin**

**Moderator:
Mr. Md Nor Ridzuan bin Daud**

**Fakulti Sains Komputer dan Teknologi Maklumat
Universiti Malaya**

ACKNOWLEDGEMENT

First of all, I would like to note that the title of my thesis was proposed by me to my supervisor. Now why would I want to do that instead of just doing the usual pick-any-title-from-any-lecturer practice? Well, suffice for me to say that I came across this title while I was browsing the internet many a months ago, and it has captivated me ever since. It has been a great pleasure indeed for me to pursue this noble goal of trying to build a face recognition system. All in all, it was a great adventure and I really enjoyed it. In that regards, I would also like to thank the people that has been involved in the process of building the face recognition system.

First and foremost, I would like to thank Assoc. Prof. Dr. Roziati Zainuddin for supervising me on my thesis work, and for accepting the proposal to my thesis title in the first place. Apart from that she also gave me a lot of ideas and motivation that really drives me to go on with the project. I would like to extend my thanks also to Mr. Md Nor Ridzuan bin Daud, for being my moderator.

Secondly, I would like to thank my friends and family members who have always been there for me in times of despair. Needless to say, there were also there when I was jovial and all. My friend's contribution to this thesis of mine is mostly in the form of discussions, where we would all sit down and discuss about the matter related to our

thesis work, while enjoying our cups of teh tarik. As for my family, just the thought of them gives me that extra strength to carry on with the project.

Last but most important of all, I would like to forward my thanks and praises to Allah, who has always watched over me day and night. May His light shine upon us all, and make us a better person than whom we are today. Amin.

ABSTRACT

Biometric is an area which deals with that of recognition processes, whether it be that of the iris scan, fingerprints, retina scan, voice recognition and face recognition. As for this thesis, the title is face recognition, which deals with the recognition of faces using computational methods. The algorithm that was adopted for this particular project is the eigenface algorithm, which preprocesses a set of face images into eigenvectors, or eigenfaces as some might like to call it, which is actually a face space that looks like that of ghostly faces, that shall be used as the base for the recognition process. After the pre-processing has been done, a single face image file is fed to the system and pre-processed into an eigenface, and that eigenface shall be mapped to the eigenfaces of the set of face images, and recognition is done, by calculating the similarities between them. The result of the recognition process is that, if the face is classified as a known face, then it will be displayed, and if it is not a known face, then it will not be displayed.

TABLE OF CONTENTS

ACKNOWLEDGEMENT.....	ii
ABSTRACT.....	iv
TABLE OF CONTENTS.....	v
LIST OF FIGURES.....	ix
LIST OF TABLES.....	xii
CHAPTER 1: INTRODUCTION.....	1
1.1 PROJECT OVERVIEW.....	1
1.2 PROJECT MOTIVATION.....	2
1.3 OBJECTIVE OF PROJECT.....	3
1.4 SCOPE OF THE PROJECT.....	3
1.5 EXPECTED OUTCOME.....	5
1.6 PROJECT SCHEDULE.....	5
1.7 REPORT LAYOUT.....	6
1.8 CHAPTER SUMMARY.....	7
CHAPTER 2: LITERATURE REVIEW.....	8
2.1 INTRODUCTION.....	8
2.2 DOMAIN STUDIES.....	8

2.2.1 DEFINATION.....	9
2.2.2 CHALLENGES ASSOCIATED WITH FACE RECOGNITION.....	9
2.2.3 METHODS IN FACE RECOGNITION.....	10
2.2.3.1 NEURAL NETWORK.....	12
2.2.3.2 EIGENFACE.....	15
2.2.3.3 TEMPLATE MATCHING.....	17
2.2.3.4 ELASTIC BUNCH GRAPH MATCHING.....	19
2.2.3.5 FISHERFACE.....	21
2.2.4 EXISTING SYSTEM REVIEW.....	23
2.3 TECHNOLOGY REVIEW.....	32
2.3.1 DEVELOPMENT METHODS.....	32
2.3.2 OPERATING SYSTEM.....	37
2.3.3 PROGRAMMING LANGUAGE.....	41
2.4 CHAPTER SUMMARY.....	46
CHAPTER 3: METHODOLOGY.....	47
3.1 SOFTWARE DEVELOPMENT PROCESS.....	47
3.2 METHODOLOGIES CONSIDERATION.....	48
3.2.1 BENEFITS OF A GOOD METHODOLOGY.....	48
3.2.2 CONCLUSIONS ON DEVELOPMENT METHODOLOGY.....	49
3.2.2.1 OVERVIEW OF UNIFIED PROCESS (UP).....	50
3.2.2.2 THE THREE UNIQUE CHARACTERISTICS OF UNIFIED PROCESS.....	56

3.2.2.3 PRIMARY MODELS OF THE UNIFIED PROCESS.....	64
3.2.2.4 JUSTIFICATION OF METHODOLOGY.....	65
3.3 UNIFIED MODELLING LANGUAGE.....	66
3.3.1 UNIFIED MODELLING LANGUAGE DIAGRAMS.....	67
3.4 INFORMATION GATHERING METHODS.....	73
3.5 CONCLUSIONS ON TOOLS AND TECHNOLOGY.....	74
3.6 CHAPTER SUMMARY.....	75
CHAPTER 4: SYSTEM ANALYSIS.....	76
4.1 TECHNIQUES FOR REQUIREMENT ELICITATION.....	76
4.2 SYSTEM REQUIREMENT ANALYSIS.....	77
4.2.1 FUNCTIONAL REQUIREMENTS.....	77
4.2.2 NON-FUNCTIONAL REQUIREMENTS.....	78
4.3 PROPOSED ALGORITHM.....	78
4.3.1 JUSTIFICATION OF CHOSEN TECHNIQUE.....	79
4.3.2 OVERVIEW OF PROPOSED ALGORITHM.....	79
4.4 HARDWARE REQUIREMENT.....	85
4.5 SOFTWARE REQUIREMENT.....	85
4.6 FACE DATABASE.....	86
4.7 CHAPTER SUMMARY.....	86
CHAPTER 5: SYSTEM DESIGN.....	87
5.1 DESIGN OF THE SYSTEM FUNCTIONS.....	87

5.2 FACE RECOGNITION ALGORITHM DESIGN.....91

5.3 USER INTERFACE.....93

 5.3.1 ADOPTED PRINCIPLES.....93

 5.3.2 USER INTERFACE FOR FACE RECOGNITON SYSTEM.....95

5.4 CHAPTER SUMMARY.....96

CHAPTER 6: SYSTEM IMPLEMENTATION.....92

6.1 PROCESSING THE INPUT IMAGE.....92

6.2 IMPLEMENTATION OF ALGORITHM.....98

 6.2.1 FACE PREPROCESSING.....99

 6.2.2 FACE RECOGNITION / MATCHING.....101

6.3 IMPLEMENTATION OF THE SYSTEM.....102

6.4 DISCUSSION.....104

6.5 CHAPTER SUMMARY.....104

CHAPTER 7: TESTING.....105

7.1 UNIT TESTING.....105

7.2 INTEGRATION TESTING.....107

7.3 SYSTEM TESTING.....108

 7.3.1 FUNCTION TESTING.....108

 7.3.2 PERFORMANCE TESTING.....108

7.4 DISCUSSION.....114

7.5 CHAPTER SUMMARY.....115

CHAPTER 8: SYSTEM EVALUATION.....116

8.1 RESULTS..... 116

8.2 STRENGTHS AND WEAKNESSES.....118

8.3 FUTURE APPLICATION.....119

8.4 CHAPTER SUMMARY.....120

APPENDIX.....121

APPENDIX A. FACE DATABASE.....121

APPENDIX B. USER MANUAL.....122

REFERENCES.....137

LIST OF FIGURES

Figure 1.1: Project Schedule

Figure 2.1: Neural network structure

Figure 2.2: Eigenvectores, or eigenfaces, as they are termed

Figure 2.3: Elastic Graph Matching example

Figure 2.4: Fisher's Linear Discriminant, shown here, seperating different classes from one another.

Figure 2.5: FACEPass Engine

Figure 2.6: FACEPass access attempts

Figure 2.7: Standard Eigenfaces

Figure 2.8: MIT Media Lab Database Photobook

Figure 2.9: The Yale Database that was used.

Figure 2.10: The Waterfall Process

Figure 2.11: The Spiral Model

Figure 2.12: The Unified Software Development Process

Figure 2.13: Windows XP Professional interface

Figure 2.14: SuSE Linux 9.1 distribution interface

Figure 3.1: The Architecture of the Unified Process

Figure 3.2: The Primary Models of the Unified Process.

Figure 3.3: Class Diagrams

Figure 3.4: Object Diagrams

Figure 3.5: Use Case Diagrams

Figure 3.6: Sequence Diagrams

Figure 3.7: Collaboration Diagrams

Figure 3.8: Statechart Diagrams

Figure 3.9: Activity Diagrams

Figure 3.10: Component Diagrams

Figure 3.11: Deployment Diagrams

Figure 4.1: The Face Recognition System Work Flow

Figure 4.2: The mean of an image.

Figure 4.3: These are examples of the eigenfaces set of images

Figure 5.1: Sequence Diagram for Face Recognition Process

Figure 5.2: Sequence Diagram for Pre-Processing of Face Images Process

Figure 5.3: Activity Diagram for Face Recognition Process

Figure 5.4: User Interface for the Face Recognition system.

Figure 8.1: Recognition of Face

Table 7.2: Similarity values for various combinations of images and detection values

Table 7.3: Detection and response time

Table 7.4: Detection range of 1-10 and number of matches made by the system

LIST OF TABLES

Table 1.1: The Gantt chart Schedule for the Project

Table 4.1: Hardware Specification.

Table 7.1: Recognition Rates for various combinations of images and division values.

Table 7.2: MinEigenDiff values for various combinations of images and division values.

Table 7.3: Divisions and response time

Table 7.4: Division range of 1-10 and number of right matches made by the system

CHAPTER 1

INTRODUCTION

Biometric is an area that is continuously being studied now and then by researchers worldwide. The word “Bio” in biometric, stands for “life”, and “metric”, is “to measure”. Biometric has been used since the prehistoric times. It was and still is the most basic of ways to recognize and measure something. For example, the Chinese has been using fingerprints since the 14th century for identification, and in the 17th century, it was used to seal official documents. There are many general definitions for the word biometric, but since this is a computer related project, thus the definition of biometric must be in a more computer related manner. With that, the definition of biometric is using a computer to recognize physical and behavior characteristics of a human. These traits are then captured by a computer and used to make a one-to-one authentication and one-to-many comparison based on unique features. Today, they are many type of biometrics, namely finger prints, retina scan, iris scan, voice recognition and last but not least, face recognition, which is the topic of this project.

1.1 Project Overview

Face recognition is an important subject in the field of computer vision systems. Although there are already many people doing face recognition systems, the fact is that face recognition is still at its infancy and it is an open and active area of research. There are plethoras of methods that can be used to build a face recognition system. For this

particular face recognition system that I will be building for this project of mine, I shall be implementing the eigenface method. The eigenface approach is actually a principal component analysis method, in which a small set of characteristic pictures are used to show the variation between face images. These are actually the eigenvectors (eigenfaces) of the covariance matrix for the set of face images. The eigenvectors are then used to define the subspace of face images that is also known as face space. As for the recognition part, a new image is projected into the subspace spanned by the eigenvectors (eigenfaces) and then classifying them by comparing its position in the face space with the positions of known individuals. If the face image resembles as close as possible to the images of the known individuals, then it will be classified as a “known” face image, and it will be classified as “unknown” face image if otherwise.

1.2 Project Motivation

The main reason for pursuing this project is due to the fact that face recognition is not offered as a subject at the faculty. Thus, this is good enough an opportunity to learn more about this technology, and treat it as a whole new subject on its own. Another reason would be to study other forms of security measures that could be implemented using non intrusive methods such as face recognition itself. As we are all aware, most of the types of security related to the cyber world have been provided by way of encryption. The question is how secure is it to rely upon encryption alone? Nowadays more and more methods of breaking encryptions have been built and hackers and crackers alike are getting smarter everyday. For these reasons, we ought to seek out new ways to confront these threats to security, namely by using biometrics such as face

recognition. Other than that, it is just an interest in wanting to know about the technology behind face recognition itself, and what it is all about, and to challenge myself into building one face recognition system.

1.3 Objective of Project

- To study and review the entire methodology for face recognition and also to implement the algorithm for the system itself.
- To choose either one or a combination of technique in building a workable software program what is usable.
- To build a face recognition software that can be used by implementing the methodology and algorithm as stated in the previous objectives.

1.4 Scope of Project

The project will be about building a workable face recognition software, in which the software will work in such a way:

- Firstly, a number of faces with similar characteristics will be grouped together and the covariance of the face images shall be calculated, and the calculated result is known as eigenvectors, and since it resembles closely to that of the face images, thus it is also known as eigenfaces.
- A new face image is then fed through the system and using the eigenfaces as comparisons to the new face image, the new face image will then be classified as

a known image if it resembles very close to the eigenfaces, and as an unknown face image if it doesn't.

Apart from that, it must also be stated that this project is strictly for educational purposes only, and that the software, too, is for that to be used for educational purposes. Thus, the software might not be the same or at the same level as that of the more commercial face recognition software. The performance of the software is also not judged by its execution time or anything similar to that. The main thing about this project is to get the software to run and execute accordingly as planned.

The term face recognition is also always mistaken with face detection and other intelligent computer vision related topics. As for this is a face recognition project, it should be noted here that there is no detection being done in the process, for all the images already contain faces and the size of the images are about the same scale.

Another one thing that should be noted is that the software doesn't recognize faces in dynamic images, such as video and such. The system only recognizes faces in static images, and therefore, only static images are being used by the system.

1.5 Expected Outcome

The software is expected to do what it was meant to do; firstly it is to form a singular value decomposition of the set of face images, and the vectors produced by it are called eigenfaces. After that, each of the new face images that is being fed through the system are being compared to the eigenfaces in order to recognize each and every face image as a known face image or as an unknown face image.

1.6 Project Schedule

The project took a span of 2 semesters to be completed, in which the first part of the semester is to research on the topic that was chosen. For the second part of the semester, the coding, building and implementation of the system takes place.

ID		Task Name	Duration	Start	Finish	Predecessors
1		Literature Review	30 days	Tue 6/1/04	Mon 7/12/04	
2		Methodology Research	15 days	Tue 6/22/04	Mon 7/12/04	
3		Inception	15 days	Tue 7/13/04	Mon 8/2/04	2
4		Life-Cycle Objectives	0 days	Mon 8/2/04	Mon 8/2/04	3
5		Elaboration (first iteration)	18 days	Tue 8/3/04	Thu 8/26/04	4
6		Elaboration (second iteration)	15 days	Fri 8/27/04	Thu 9/16/04	5
7		Life-Cycle Architecture Milestone	0 days	Thu 9/16/04	Thu 9/16/04	6
8		Construction (first iteration)	78 days	Fri 9/17/04	Tue 1/4/05	7
9		Construction (second iteration)	67 days	Wed 1/5/05	Thu 4/7/05	8
10		Initial Operation Capability	0 days	Thu 4/7/05	Thu 4/7/05	9
11		Transition	10 days	Fri 4/8/05	Thu 4/21/05	10
12		Product Release Milestone	0 days	Thu 4/21/05	Thu 4/21/05	11
13		Documentation	234 days	Tue 6/1/04	Fri 4/22/05	

Table 1.1: The Gantt chart Schedule for the Project

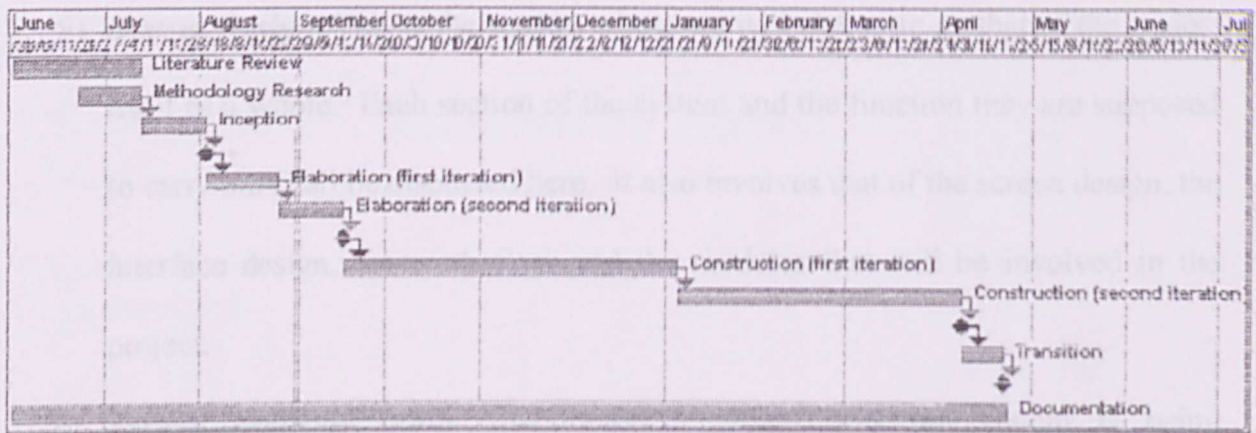


Figure 1.1. Project Schedule

1.7 Report Layout

The report is being partitioned into 8 parts, where each of the part takes off from the part before that. The list of the report layout is as such:

- 1) **Introduction:** This where we explain briefly about the system that we are about to build, the scope of the project, and the objective of the project, its timeline, and its expected outcome.
- 2) **Literature Review:** In this section, we discuss the problems or limitations associated with the project before it is being carried out. In this section we will also have a look at systems that has been built before this and the techniques or methods that they have used in building their systems.
- 3) **Methodology:** An in-depth elaboration regarding the methods of research and the techniques that will be used in building the system.
- 4) **System Analysis:** Here we shall discuss the requirements that are associated with the project, such as the non functional requirements and the functional requirements, hardwares and softwares that will be used in this project.

- 5) **System Design:** This is the section where we will elaborate on that of the project itself as a whole. Each section of the system and the function they are supposed to carry out shall be discussed here. It also involves that of the screen design, the interface design, the work flow and the modules that will be involved in the project.
- 6) **System Implementation:** Elaboration regarding how the system is being implemented and this refers to the modules and algorithm that has been change from a mere idea to that of a true working code by using any programming language that is most suitable for the project.
- 7) **Testing:** This section discusses about how the systems works out, whether it works as expected or otherwise.
- 8) **System Evaluation and Conclusion:** The results of the system, problems and ways to go about it, advantages and disadvantages, and any future plans for the software, suggestions and conclusions shall be discussed here.

1.8 Chapter Summary

Face recognition has indeed a long way to go, but at the same time, that leaves us a lot of space to venture and discover about this field. In this chapter we have discussed about face recognition rather briefly, about how the system is going to be built, roughly how it is going to work, about the project motivation, the scope of the project, the schedule of the project and its expected outcome.

CHAPTER 2

LITERATURE REVIEW

2.1 Introduction

The main goal of this chapter is to provide a thorough understanding about the matter at hand, and here, the matter at hand is about face recognition. It is perceived as a pre-analysis phase where different methods of face recognition applied by accredited scholars and researchers are being studied and researched. Studies are also done upon existing softwares that are related to face recognition. At the end of the chapter, we shall review that of about the technology that will be used to build the face recognition software.

2.2 Domain Studies

In a vision based human-machine/computer interaction, the one thing that is of the utmost importance is the image that contains faces. Many researches has been carried out to find the most suitable algorithm or method to that of face recognition, but like so many other things in life, no one algorithm or method is 100% perfect, for each has its own advantages and disadvantages. Although every face recognition system differs slightly from each other, the same routine applies throughout the flow of the system. Firstly, all the pictures shall be fed into the machine so that it could be train to know how to differentiate faces and non-faces images. Secondly, an unknown face image is

being fed through the system and then the system shall identify the face as a known face image, a non face image or as an unknown face image depending upon their algorithm and methods that they have applied. Before we go into the details, here we shall have a look at the few definitions of words that shall be used throughout the project.

2.2.1 Definition

- **Face:** The face is the frontal view of the head.
- **Face Analysis:** The study of the face itself.
- **Face Recognition:** Match the given face to a database of known faces.
- **Face Detection:** Find a face within a given scene/image.
- **Face Localization:** Find the face given that a single face is known to appear in the image.
- **Facial Expression:** The condition of the face that is being showed as for to show emotion (e.g. sadness, happiness, depressive).

2.2.2 Challenges Associated with Face Recognition

In the process of building any types of softwares, there are many challenges that have to be encountered before building the software itself. Below, we shall list the types of challenges or problems that are known or may arise. The FERET test that was done not so long ago (1994) showed that in face recognition, there are at least two major challenges, which are:

- ***Illumination Variation:*** Images of the same face appear differently due to the change in lighting. The solution proposed so far is to discard the first few eigenfaces from the set of eigenfaces.
- ***Pose Variation:*** Images of the same face that appear slightly different from one another because of how the image was captured, whether from the side, front, or slightly to the left or right of the person's face. Pose variation also applies to how the person's face was when the image was captured (e.g. Smiling, frowning, etc.).

2.2.3 Methods in Face Recognition

Basically, there are two classes of techniques to that of face recognition: image face recognition and video face recognition. Image face recognition is done by using a static image capture using a web cam, digital camera, or the likes of it, while video based recognition is done by using video capture from a video, whether it be static or dynamic. Below are the details about that of the two techniques:

Image based face recognition

- Holistic matching methods
- Feature-based (structural) matching methods
- Hybrid methods
- Summary and research directions

Video based face recognition

- Still-image methods
- Multi-modal methods
- Spatio-temporal methods
- Summary and research directions

Since this project does not use anything related to video capture, we wouldn't be discussing that any further. Instead, we are going to concentrate more on that of the image based face recognition techniques. The most widely used techniques amongst the four techniques in image based face recognition are the Holistic Matching methods and the Feature-Based matching methods. Here we shall list a few of the more widely used approach for these two techniques, namely the Neural Network approach, Template matching approach, Elastic Graph Matching approach, Fisherface approach and the Eigenface approach.

2.2.3.1 Neural Network

Neural Networks are relatively crude electronic models based on the neural structure of the brain. The brain basically learns from experience. It is natural proof that some problems that are beyond the scope of current computers are indeed solvable by small energy efficient packages. This brain modeling also promises a less technical way to develop machine solutions. This new approach to computing also provides a more graceful degradation during system overload than its more traditional counterparts.

These biologically inspired methods of computing are thought to be the next major advancement in the computing industry. Even simple animal brains are capable of functions that are currently impossible for computers. Computers do rote things well, like keeping ledgers or performing complex math. But computers have trouble recognizing even simple patterns much less generalizing those patterns of the past into actions of the future.

Now, advances in biological research promise an initial understanding of the natural thinking mechanism. This research shows that brains store information as patterns. Some of these patterns are very complicated and allow us the ability to recognize individual faces from many different angles. This process of storing information as patterns, utilizing those patterns, and then solving problems encompasses a new field in computing. This field, as mentioned before, does not utilize traditional programming but involves the creation of massively parallel networks and the training of those networks to solve specific problems. This field also utilizes words very different from traditional computing, words like behave, react, self-organize, learn, generalize, and forget.

There are many implementation of neural network to that of face recognition, but the one that shall be used as an example here shall be from the work of Qing Jiang, which is titled “Principal Component Analysis and Neural Network Based Face Recognition “[1].

The neural network has a general back-propagation structure with three layers. Assuming that the input layer has 100 nodes from the new face descriptors, the hidden layer has 10 nodes and the output unit gives a result from 0.0 to 1.0 telling how much the input face can be thought as the network's host.

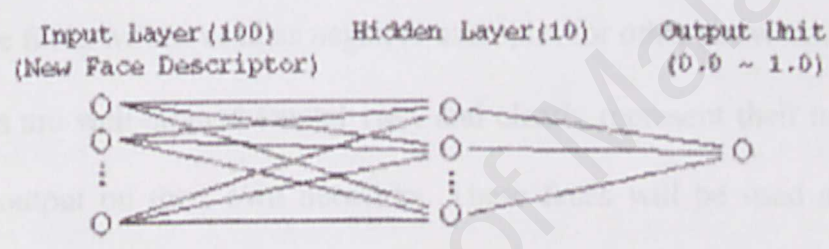


Figure 2.1: Neural network structure

Each neural net identifies whether the input face is the network's host or not. The recognition algorithm selects the network with the maximum output. If the output of the selected network passes a predefined threshold, it will be reported as the host of the input face. Otherwise the input face will be rejected.

After the neural network structure was set, the most important thing was to prepare the training examples. In the beginning of the training, a few numbers of face images were selected from each person that was well aligned frontal view. Any of them could represent their host clearly.

After the basic neural networks were created, they are being run over new faces from the individuals in the database. If the image failed to pass the face detection test, it will be ignored. If the face detection code reports a face in the image, it will be applied to the face recognition code. The system checks the recognition result to find more faces for training. Here each face will fall into one of the four categories as following:

- Faces have high output on their own networks and low output on other networks.
No action here.
- Faces have high output on both their own networks and some other networks.
These faces will be used as negative examples for other networks.
- Faces are well-aligned frontal view and clearly represent their hosts. They have low output on their own networks. These faces will be used as both positive examples for their own networks and negative examples for other networks.
- Faces are not well cut and can't represent their hosts clearly. They have low output on their own networks. If they have high output on some other networks, they will be included as negative examples for other networks. Otherwise they will be ignored.

Once we get these new faces, they are added to training examples and the neural network is retrained. Recognition errors will be corrected and the total performance will be improved. While adding some examples from a specific individual will improve the performance of the network, it also influenced the performance of other networks.

If the face is recognized as one in the database, the speaker will say his name, otherwise it will say "no match". The experiments were conducted under roughly the same

illumination condition. Wide expression variations were incorporated. The faces were basically frontal view without any significant orientation change.

The neural network model is a good model to be implemented into the face recognition system. But nonetheless, there is one slight disadvantage regarding it: Neural network models tends to take a toll on the training time, thus their performance are slower than that of the other methods.

2.2.3.2 Eigenface

This is one the more famous methods related to face recognition, and it was first introduced by Matthew A. Turk and Alex P. Pentland, which is titled "Face Recognition Using Eigenfaces"[2]. The eigenface representation method for face recognition is based on the principal component analysis. The main idea is to decompose face images into a small set of characteristic feature images called eigenfaces, which may be thought of as the principal components of the original images. These eigenfaces function as the orthogonal basis vectors of a linear subspace called "face space". Recognition is performed by projecting a new face image into this face space and then comparing its position in the face space with those of known faces.



Figure 2.2: Eigenvectors, or eigenfaces, as they are termed.

Basically, the workflow of the system is as such:

- Form a face library that consists of the face images of known individuals.
- Choose a training set that includes a number of images (M) for each person with some variation in expression and in the lighting.
- Calculate the $M \times M$ matrix L , find its eigenvectors and eigenvalues, and choose the M' eigenvectors with the highest associated eigenvalues.
- Combine the normalized training set of images to produce M' eigenfaces.
- Store these eigenfaces for later use.

- For each member in the face library, compute and store a feature vector.
- Choose a threshold e that defines the maximum allowable distance from any face class. Optionally choose a threshold f that defines the maximum allowable distance from face space.
- For each new face image to be identified, calculate its feature vector and compare it with the stored feature vectors of the face library members.
- If the comparison satisfies the threshold for at least one member, then classify this face image as "known", otherwise a miss has occurred and classify it as "unknown" and add this member to the face library with its feature vector.

The advantages of using eigenface are that the recognition process is faster and it is easy to implement. The down side of the eigenface process is that it is time consuming at times and the size and location of each face image must remain similar to one another.

2.2.3.3 Template Matching

Template matching has many a definition, which are:

- Technique used in classifying objects.
- Template matching techniques compare portions of images against one another.
- Sample image may be used to recognize similar objects in source image.

- If standard deviation of the template image compared to the source image is small enough, template matching may be used.
- Templates are most often used to identify printed characters, numbers, and other small, simple objects.

The most direct method used for face recognition is the matching between the test images and a set of training images based on measuring the correlation. This method is called the Template Matching method. The matching process moves the template image to all possible positions in a larger source image and computes a numerical index that indicates how well the template matches the image in that position. Matching is done on a pixel-by-pixel basis.

Basically the workflow of the template matching method is as such:

- Normalize the images
- Each person is represented by a set of 4 masks: eyes, nose, mouth and face (region from eyebrows to below chin). Their location is relative to eye position.
- Correlations are found for each feature. (all 4) The new face is matched with the highest cumulative score.
- Correlation is sensitive to illumination gradients. 4 different normalization techniques were tried and compared: Gradient information was the best.
- Recognition as a function of the image resolution is tested.
- Feature performance by mask: (decreasing order)
 1. eyes
 2. nose

3. mouth
4. whole face

- The similarity scores for each feature were added to produce the global score.

The good thing about using template matching is that it works faster than some of the other more delicate algorithms out there, and also it is simpler to implement.

Figure 2.3: Elastic Graph Matching example

2.2.3.4 Elastic Bunch Graph Matching

The other dominant trend in face recognition has been feature matching: deriving distance and position features from the placement of internal facial elements. Kanade [3] developed one of the earliest face recognition algorithms based on automatic feature detection. By localizing the corners of the eyes, nostrils, etc. in frontal views, the system computed parameters for each face, which were compared (using a Euclidean metric) against the parameters for known faces. A more recent feature-based system, based on elastic bunch graph matching, was developed by Wiskott *et al.* [4] as an extension to their original graph matching system.

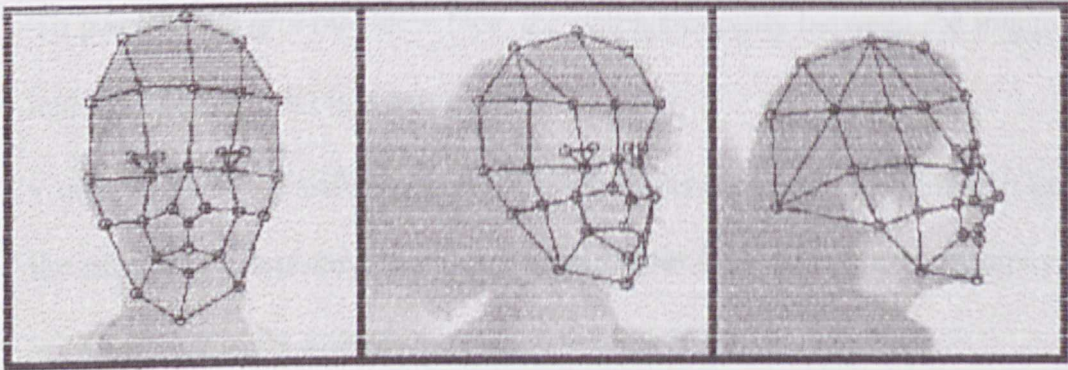


Figure 2.3: Elastic Graph Matching example

Wiskott *et al.* present a general in-class recognition method for classifying members of a known class of objects. The workflow of the system is as such:

- Faces are represented as graphs, with nodes positioned at fiducial points, and edges labeled with 2-D distance vectors. Each node contains a set of 40 complex Gabor wavelet coefficients, including both phase and magnitude, known as a *jet*.
- Wavelet coefficients are extracted using a family of Gabor kernels with 5 different spatial frequencies and 8 orientations; all kernels are normalized to be zero mean.
- To identify a new face, an object-adapted graph has to be positioned on the face using elastic bunch graph matching. This can be performed automatically if the face bunch graph (FBG) has already been initialized with enough faces (approximately 70).
- A face bunch graph (FBG) consists of a collection of individual face model graphs combined into a stack-like structure, in which each node contains the jets of all previously initialized faces from the database.

- To position the grid on a new face, the graph similarity between the image graph and the existing FBG is maximized.
- Graph similarity is defined as the average of the best possible jet match between the new image and any face stored within the FBG minus a topography term, which accounts for distortion between the image grid and the FBG.
- After the grid has been positioned on the new face, the face is identified by comparing the similarity between that face and every face stored in the FBG.

The good thing about using the method is that it is robust, and it doesn't use the whole face. Instead it only focuses on the fiducial points of the face. The method also is insensitive to various types of lighting, face position and expression.

With all the advantages, there are also disadvantages. Firstly, the method may require more computational effort than the likes of eigenfaces method or such. Another disadvantage of this technique is that the tedious grid placement must be performed manually for the first 70 images or so, before the automatic graph matching becomes sufficiently reliable.

2.2.3.5 Fisherface

Based on the work of P. Belhumeur, J. Hespanha, and D. Kriegman from the Yale University [5]. According to their research, eigenfaces attempt to maximise the scatter of the training images in face space. While the name eigenface and fisherface might have a bit of a resemblance when they are pronounced, they work in a different way

altogether. Fisherfaces attempts to maximise the between class scatter, while minimising the within class scatter. In other words, Fisherface attempts to moves images of the same face closer together, while moving images of difference faces further apart. Fisherface method works by applying the Fisher's Linear Discriminant, where it attempts to project the data in such a way that different classes are separated.

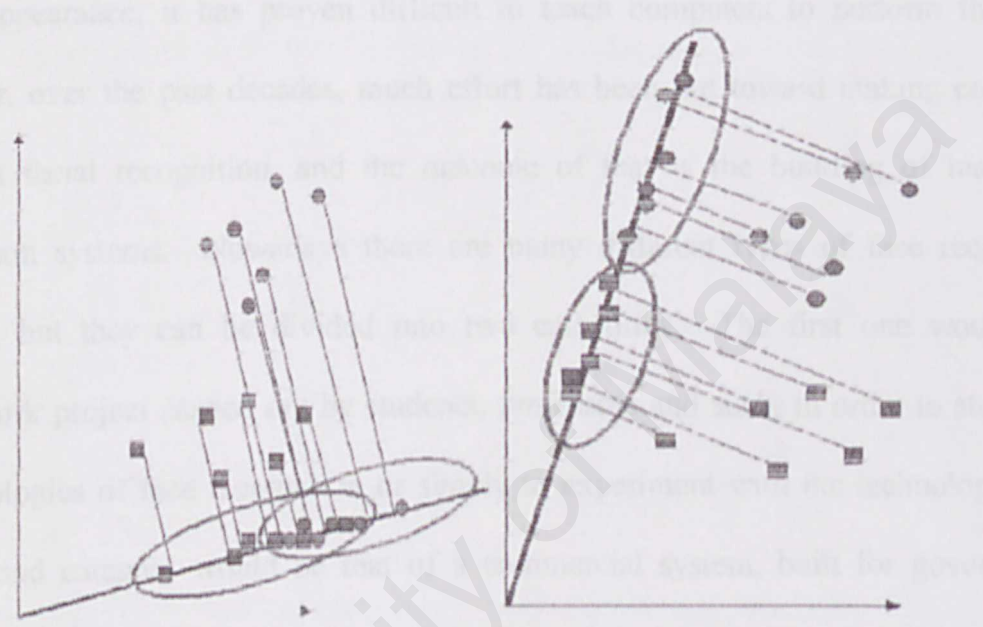


Figure 2.4: Fisher's Linear Discriminant, shown here, seperating different classes from one another.

Although all seems well, but there are a few disadvantages that has to be noted. One of the disadvantage is that although the concept very assuring, but the fact is that it is not as accurate as other biometrics. It must also be noted here that a large amount of storage is needed to store the datas. Finally, the images that are being used has to be of a high quality image, or the system will produce erroneus recognition of the face images.

2.2.4 Existing System Review

While humans have an innate ability to easily recognize the faces of other humans, even within large crowds and when factors such as emotion, facial hair, and age change the face's appearance, it has proven difficult to teach computers to perform this task. However, over the past decades, much effort has been put toward making computers better at facial recognition, and the outcome of that is the building of many face recognition systems. Nowadays there are many different types of face recognition systems, but they can be divided into two categories. The first one would be a coursework project carried out by students, professors and such, in order to study new methodologies of face recognition or simply to experiment with the technology itself. The second category would be that of a commercial system, built for governments, offices, places where security is needed. Here we shall review a number of those systems and have a look at their specifications, their methodologies, techniques, and how they fared out.

Viisage FacePASSTM

FacePASS is a face recognition enabled access control and security product used to conduct one-to-one verification of an individual's identity to authenticate that the holder of an ID document or token is the authorized user. The system carries out a real-time comparison between a live face and a stored reference image to confirm the identity of that person and grant respective rights and privileges while guaranteeing maximum

reliability, security and efficiency. It is the most effective biometric tool to secure access to physical facilities and protect sensitive areas and buildings.

In access control applications to high security areas, such as in airports, banks or nuclear power plants, FacePASS ensures that access is granted for authorized persons only. In ticketing applications for the leisure industry, e.g. zoos or fitness clubs, it effectively prevents ticket fraud with member cards or season tickets and ensures high user throughput.

FacePASS is based on Viisage's advanced face recognition technology and has been applied in daily operation in a variety of industries for several years. Product design and features have been continuously developed further thanks to the practical experience with customer installations and close interaction with users.

The combination of leading technology, self-explanatory and user-friendly system design as well as solution-oriented features provides the highest recognition performance, robustness and process speed.

All organizations and institutions that need to protect restricted areas from unauthorized entry are facing a common challenge: The fraudulent use of identities in order to claim rights of access, e.g. to access security areas. To date, the only approach to solve this problem has been a manual process of visual image comparison with security staff.

However, the effectiveness of these manual processes has been limited and inefficient. FacePASS automates this process of confirming the claimed identity and subsequently

granting permission to which the individual is entitled and offers a reliable, efficient and effective identity solution.

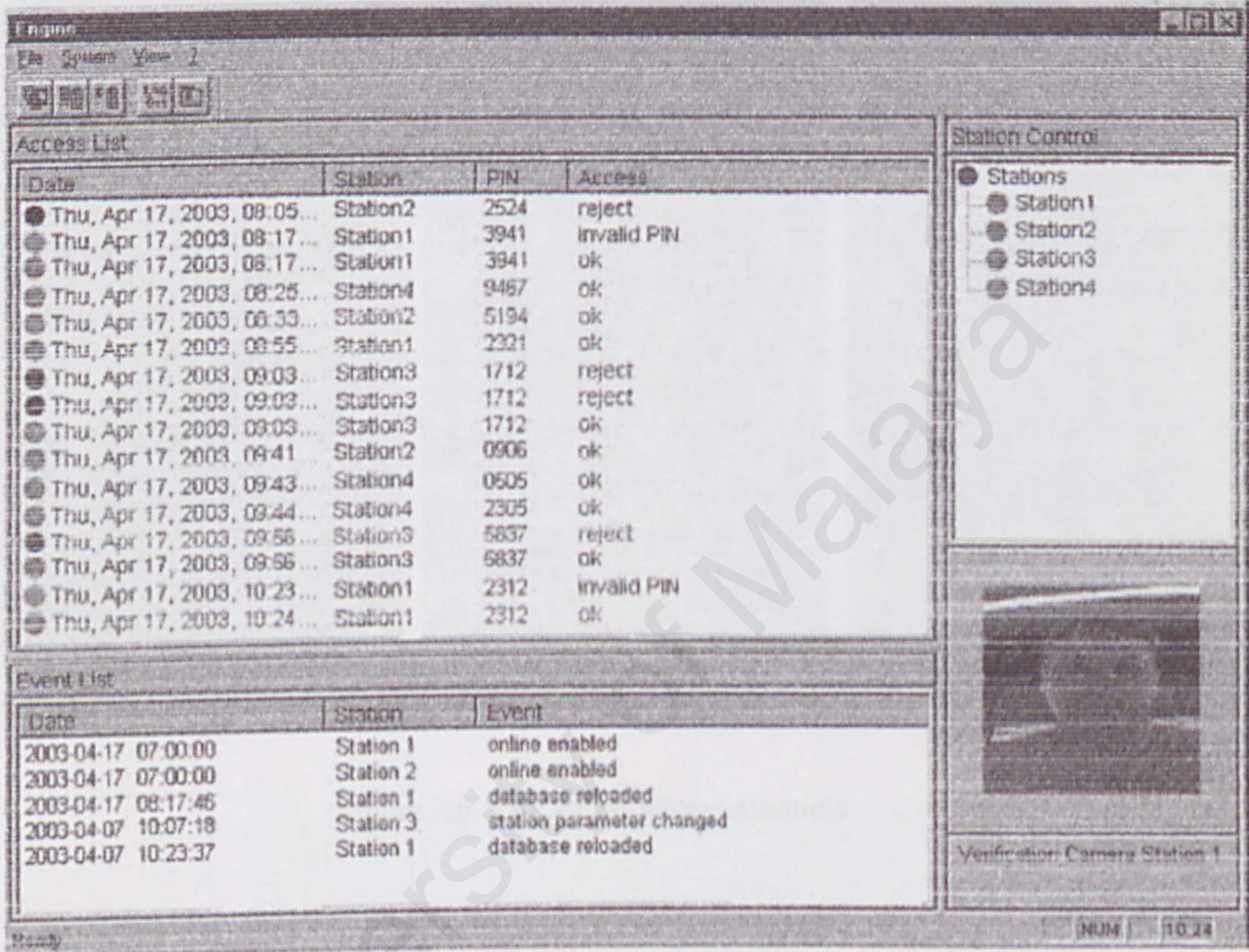


Figure 2.5: FACEPass Engine



Figure 2.6: FACEPass access attempts

It is based on Flexible Template Matching, a new and powerful technological foundation, incorporating a unique combination of multiple approaches to face recognition. Other products that has been built by the company including FacePIN, FaceFinder, etc.

Nowadays, the system is widely used in conjunction with identification cards in many US states and developing nations.

Photobook/Eigenfaces Demo

Most face recognition experiments to date have had at most a few hundred faces. Thus how face recognition performance scales with the number of faces is almost completely unknown. In order to have an estimate of the recognition performance on much larger databases, we have conducted tests on a database of 7,562 images of approximately 3,000 people.

The eigenfaces for this database were approximated using a principal components analysis on a representative sample of 128 faces. Recognition and matching was subsequently performed using the first 20 eigenvectors. In addition, each image was then annotated (by hand) as to sex, race, approximate age, facial expression, and other salient features. Almost every person has at least two images in the database; several people have many images with varying expressions, headwear, facial hair, etc.



Figure 2.7: Standard Eigenfaces

This database can be interactively searched using an X-windows browsing tool called Photobook. The user begins by selecting the types of faces they wish to examine; e.g., senior Caucasian males with mustaches, or adult Hispanic females with hats. This subset selection is accomplished using an object-oriented database to search through the face image annotations. Photobook then presents the user with the top matches found in the database. The remainder of the database images can be viewed by "paging" through the set of images. At any time the user can select a face from among those presented, and Photobook will then use the eigenvector description of that face to sort the entire set of faces in terms of their similarity to the selected face. Photobook then re-presents the user with the face images, now sorted by similarity to the selected face.

The figure below shows the typical results of Photobook similarity search using the eigenvector descriptors. The face at the upper left of each set of images was selected by the user; the remainders of the faces are the 15 most-similar faces from among the entire 7,562 images (in this case they all belong to the same individual). Similarity decreases left to right, top to bottom. The entire searching and sorting operation takes less than one second on a standard Sun Sparcstation, because each face is described using only a very small number of eigenvector coefficients. Of particular importance is the ability to find the same person despite wide variations in expression and variations such as presence of eye glasses, etc.

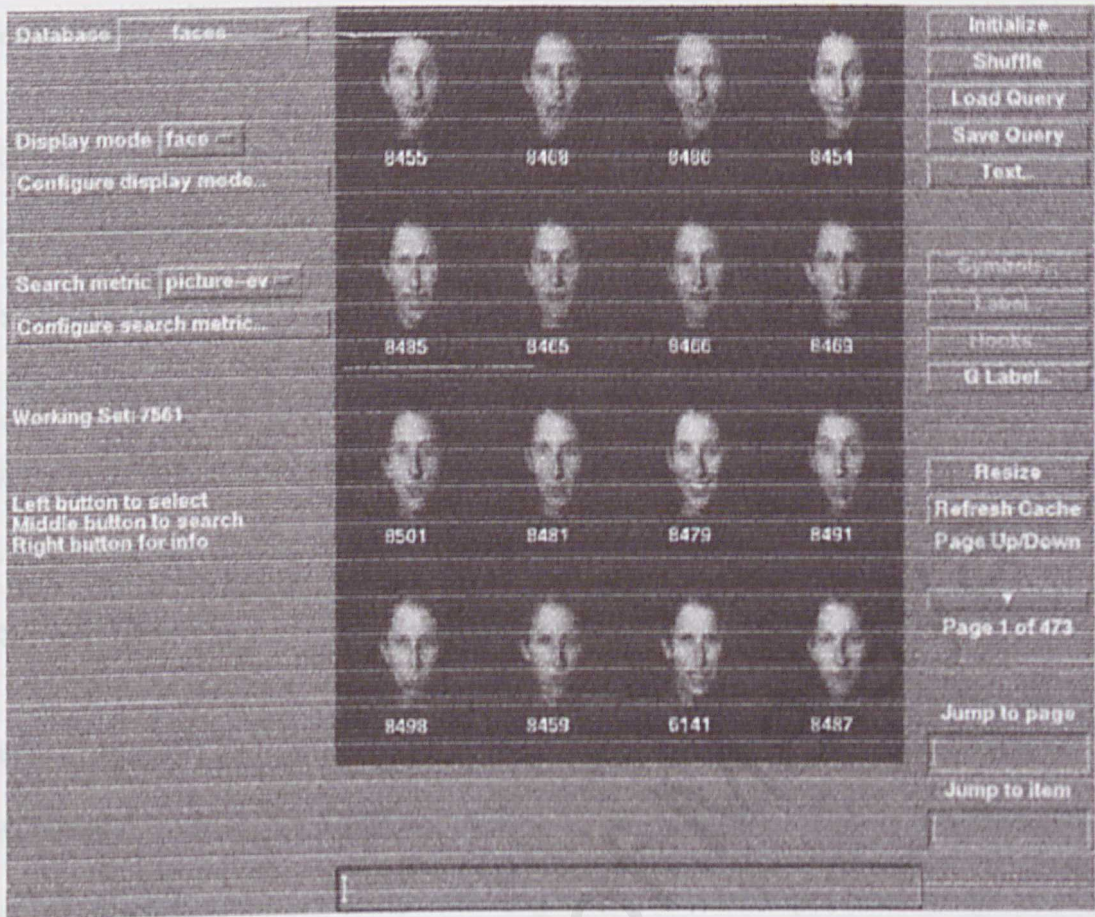


Figure 2.8: MIT Media Lab Database Photobook

To assess the average recognition rate, 200 faces were selected at random, and a nearest-neighbor rule was used to find the most-similar face from the entire database. If the most-similar face was of the same person then a correct recognition was scored. In this experiment the eigenvector-based recognition system produced a recognition accuracy of 95%.

Miros (eTrue) TrueFace

TrueFace is based on neural network technology, which mimics the way the brain functions, isolating the face while ignoring the background elements. Enrollment into the system is easy; it's just like having your picture taken and can be completed by the user right on their computer. With the added security of a facial recognition logon, systems that were once simply protected by passwords can now be secured by the flash of a smile. TrueFace frees the user from having to remember multiple PINs and allows access with something that you literally cannot leave home without.

TrueFace Application Offers Identification as Well as Verification of ID Miros TrueFace face recognition product features two functionalities. It can verify that a person is who they claim to be by matching their face with a previous image. This takes about a second. TrueFace also can identify a person's face by searching an existing database for a match. This can be done at a rate of 200 faces per second on a desktop workstation.

TrueFace has been shown to correctly verify greater than 99 per cent of legitimate faces for applications such as entering secure buildings. It has the same degree of accuracy for catching frauds in applications requiring human back-up, such as verifying passports. Human back-up is used when an application requires exceptions to be handled. TrueFace provides this high level of accuracy while at the same time maximizing the convenience and speed of the security process.

Face Recognition Using Kernel Methods

Eigenface or Principal Component Analysis (PCA) methods have demonstrated their success in face recognition, detection, and tracking. The representation in PCA is based on the second order statistics of the image set, and does not address higher order statistical dependencies such as the relationships among three or more pixels.

Recently Higher Order Statistics (HOS) have been used as a more informative low dimensional representation than PCA for face and vehicle detection. This particular project investigates a generalization of PCA, Kernel Principal Component Analysis (Kernel PCA), for learning low dimensional representations in the context of face recognition.

In contrast to HOS, Kernel PCA computes the higher order statistics without the combinatorial explosion of time and memory complexity. While PCA aims to find a second order correlation of patterns, Kernel PCA provides a replacement which takes into account higher order correlations.

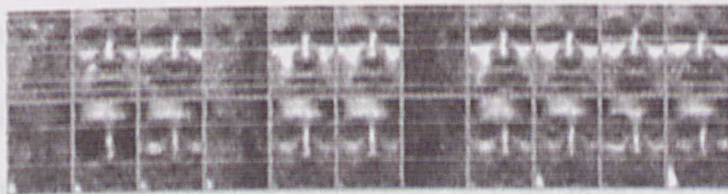


Figure 2.9: The Yale Database that was used.

Comparisons are made for the recognition results using kernel methods with Eigenface methods on two benchmarks. Empirical results show that Kernel PCA outperforms the Eigenface method in face recognition.

2.3 Technology Review

2.3.1 Development Methods

A software development method is a well-organized activities of related methods that addresses who does what activities and how, when, why, and where these activities should be done to develop a system. Although there are many software development methods, here, we shall review 3 of the better known methods available; the spiral method, the waterfall method and the Unified Process method.

The Waterfall model

The Waterfall model and evolutionary development are widely used for practical systems development. The waterfall model consists of a number of stages. These stages can be characterized and divided up in different ways, including the following:

- **Project planning, feasibility study:** Establishes a high-level view of the intended project and determines its goals.
- **Systems analysis, requirements definition:** Refines project goals into defined functions and operation of the intended application. Analyzes end-user information needs.

- **Systems design:** Describes desired features and operations in detail, including screen layouts, business rules, process diagrams, pseudo code and other documentation.
- **Implementation:** The real code is written here.
- **Integration and testing:** Brings all the pieces together into a special testing environment, then checks for errors, bugs and interoperability.
- **Acceptance, installation, deployment:** The final stage of initial development, where the software is put into production and runs actual business.
- **Maintenance:** What happens during the rest of the software's life: changes, correction, additions, and moves to a different computing platform and more.

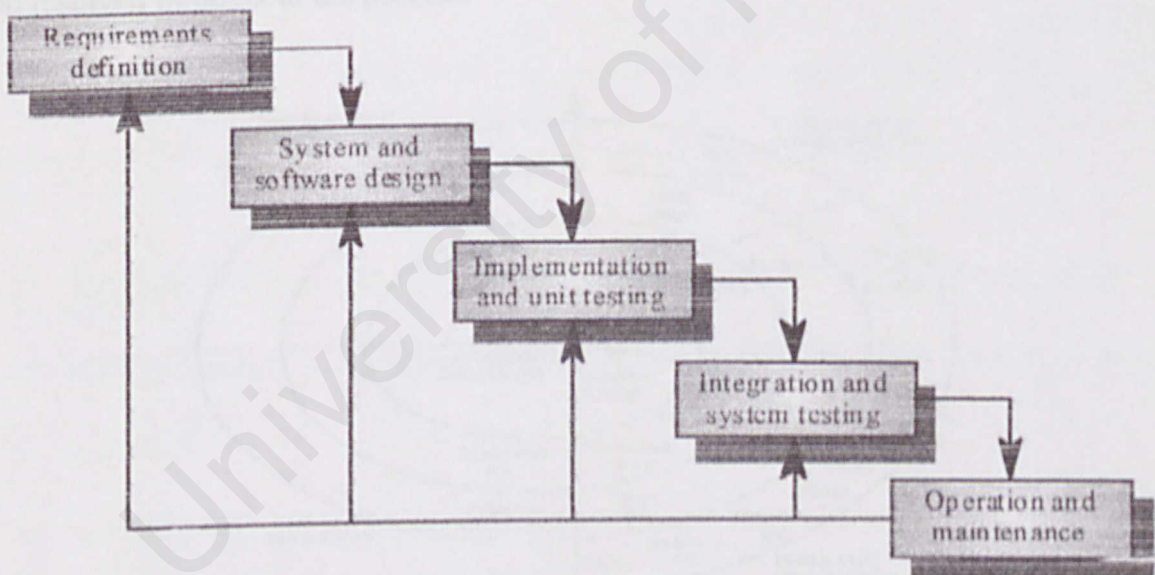


Figure 2.10: The Waterfall Process

The waterfall model is a very straight forward one, and many IT departments are still using it as their software development method. However, there are some problems associated with the waterfall model, such as stated below:

- Inflexible partitioning of the project into distinct stages
- This makes it difficult to respond to changing customer requirements
- Therefore, this model is only appropriate when the requirements are well-understood.

The Spiral model

The process in a spiral model is represented as a spiral rather than as a sequence of activities with backtracking. Each loop in the spiral represents a phase in the process. In a spiral model, there are no fixed phases such as specification or design – the loops in the spiral are chosen depending on what is required. All the risks are explicitly assessed and resolved throughout the process.

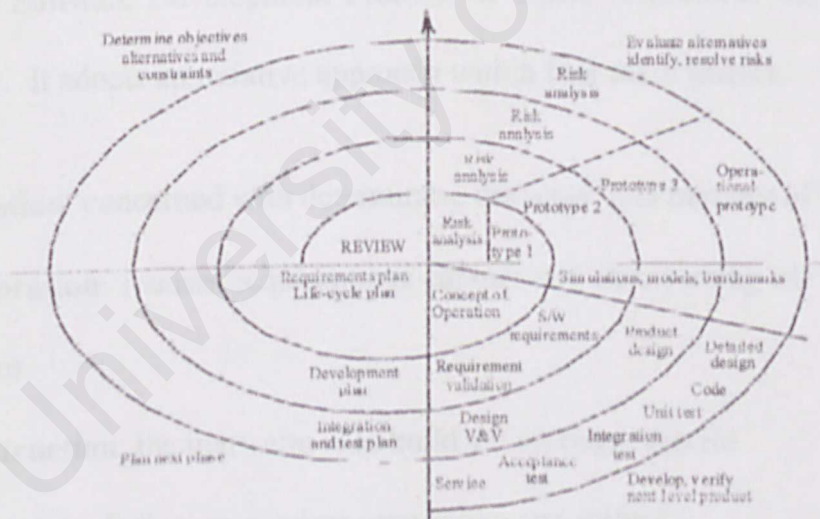


Figure 2.11: The Spiral Model

There are sectors in a spiral model and each and everyone of it has its own processes.

The sectors are as such:

- **Objective setting:** Specific objectives for the phase are identified
- **Risk assessment and reduction:** Risks are assessed and activities put in place to reduce the key risks
- **Development and validation:** A development model for the system is chosen which can be any of the generic models
- **Planning:** The project is reviewed and the next phase of the spiral is planned

Unified Software Development Process

The Unified Software Development Process, or USDP, represents the object-oriented methodology. It adopts an iterative approach within four main phases:

- **Inception:** concerned with determining the scope and purpose of the projects
- **Elaboration:** focuses requirements capture and determining the structure of the system
- **Construction:** the main aim is to build the software system
- **Transition:** deals with product installation and rollout.

Basically these are the core features of the USDP:

- Use-Case Driven
- Architecture-Centric
- Iterative and Incremental

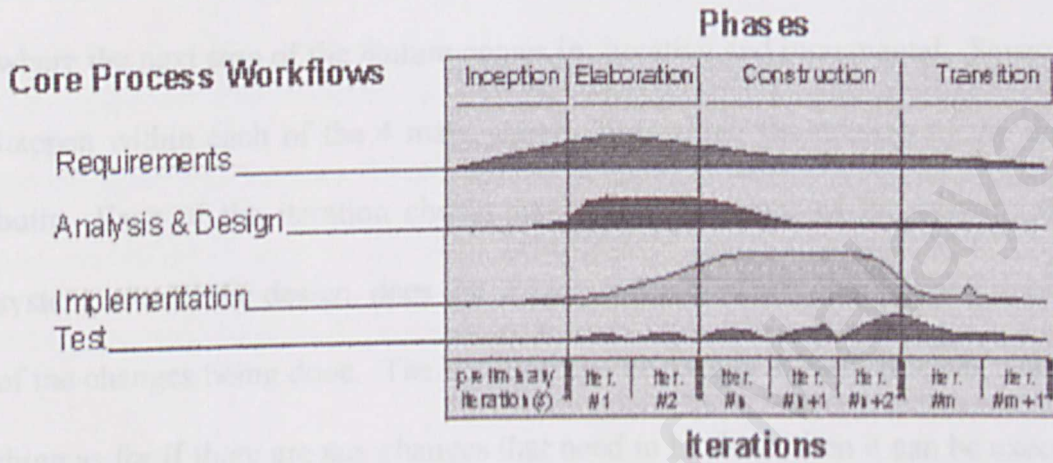


Figure 2.12: The Unified Software Development Process

The first part of the features of the USDP, stated that it is case-driven, which means that it captures the functional requirements of the software to be build. It also divides the development process into 3 smaller parts:

- Design
- Build
- Test

The second part of the USDP features, stated that it is architecture-centric, that means that after the requirements have been defined, it is then being analyzed. The analysis of the requirements are being done to get different views of the system, be it from the

design view, process, view, implementation view and the deployment view. It is being done to recognize the risk, or any side factors that has to be considered in the process and such. In other words, we can say that the software system is being developed around architecture.

After that have been done, then the building of the project will be carried out. This is where the next step of the feature comes in; iterative and incremental. Several iterations happen within each of the 4 main phases, throughout the process of the project being built. Each of the iteration checks out the requirements of the system, analyzes the system, checks the design, does the implementation of any changes and does the testing of the changes being done. The iteration can be termed as mini projects. This is a good thing as for if there are any changes that need to be done, then it can be executed within the period of iteration itself, and thus making the software a more error free software when it comes to deployment time.

2.3.2 Operating System

Operating system is the platform on which the face recognition system is to build on. As of right now, there are already a lot of operating systems around, and rest assured that many more will surface soon. As for this project, the choice for an operating system is vital indeed, for if the wrong operating system is chosen, then something bad might turn out such as a system failure and the likes. On the other hand, if the right operating system is chosen, the system then might just works flawlessly and smoothly indeed. Here shall be listed the few most likely to be the operating system of choice.

Windows XP Professional Edition

Windows XP looks different from previous versions, with a much cleaner appearance to the desktop. Apart from the Start button, there's little extra to confuse the newcomer. Click on Start, though, and the two-column menu that pops up is re-ordered and points the way to other changes in the operating system.

The color scheme suggests a bigger, brighter approach to PCs and this idea is strengthened by the bold icons and the way it's harder to get at the nuts and bolts. It encourages us to stay on the yellow brick road of applications and their documents, rather than to delve into the underlying code of it all.

The operating system, which supports both FAT32 and NTFS filing systems, offers a lot of extras. A new Wizard for printing that arranges photos to make the best use of expensive photo paper and a video editing applet which provides the basics for cutting and pasting digital video, are the new things to mention about indeed.

Others include easier home networking and the ability to allow a service technician to temporarily take over the user's PC to provide technical support.

It requires a substantial PC to run Windows XP. The minimum recommended is a 300MHz Pentium with 64MB memory and 1.5GB of hard drive space.

Windows XP is impressive. It appears to be more stable than before and adds welcome support for a number of extras, like digital cameras and video. It is a good upgrade from the recent version of windows indeed, and a lot more stable.

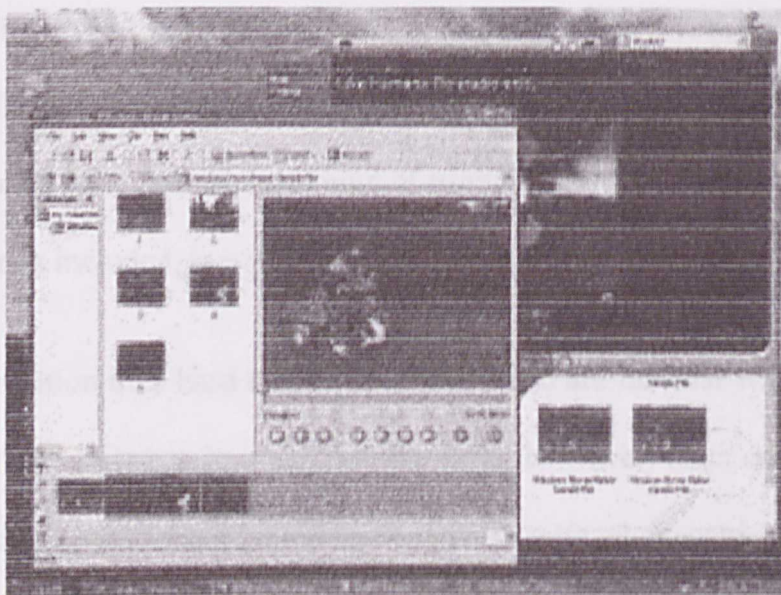


Figure 2.13: Windows XP Professional interface

SuSE Linux 9.1

SuSE Linux 9.1 comes on a two disc set, the first of which allows the user to run the operating system directly from a CD to see if the user wants to install it proper. This way, it doesn't touch the existing system.

The installation of the system only takes up about fifteen minutes once its recommendations - all of which were customizable - had been accepted. This includes establishing another partition on the main hard drive and other criteria which has to be set.

Upon reset, an option will appear allowing a choice between booting to Linux or Windows XP. Booting from here would roughly be half the time of the XP installation and the desktop screen will give a mere hint of the flexibility and options lying under the surface.

Included in this SuSE distribution was a full office suite, a powerful graphics application, solid media playback centre and an e-mail application that happily brought in Outlook Express data with the minimum of trouble. In short, everything needed to run an office system is included, in a package that is a lot cheaper than that of Windows XP.

The SuSE distribution tries hard to appeal to those who are familiar with Windows, and so its graphical user interface follows a similar style. However, there are differences. By default, there is more than one desktop layout to play with, changeable by a simple click of the mouse. Likewise, one click will also hide the menu bar, from which applications are launched. It's a fantastically intuitive interface, and one that makes a migration that bit easier.

However, Linux isn't without a couple of problems such as the detection of a sound card sometimes doesn't really work for an unknown sound card, and this is a problem indeed, for a third party driver that works with Linux has to be found, or not there would be no sound for eternity.

In Windows, there is no problem to fix such issues, yet moving over to Linux takes a little bit of learning, and just as importantly, a little bit of unlearning too. But that is not much of a problem once the user gets used to the operating system. There are other distribution of Linux around, namely Red Hat Linux, Mandrake and such, but the fact is that SuSE Linux is the easiest to start off with. Moreover, it operates in a similar fashion as that of a Windows operating system, and it is stable. If a new user wants to use an operating system other than that of the Windows family, then SuSE Linux is the operating system to be considered indeed.

3.2.2.1 Overview of the Unified Process

The Unified Process fits the general definition of a process: a set of activities that a team performs to transform a set of customer requirements into a software system. However, the Unified Process is also a generic process framework that people can customize by adding and removing activities based on the particular needs and available resources for a project. The Rational Unified Process (RUP) is an example of a specialized version of the Unified Process that adds elements to the generic framework.

The Unified Process makes extensive use of the Unified Modeling Language (UML). At the core of the UML is the model, which in the context of a software development process is a simplification of reality that helps the project team understand certain aspects of the complexity inherent in software.

The UML was designed to help the participants in software development efforts build models that enable the team to visualize the system, specify the structure and behavior of that system, construct the system, and document the decisions made along the way. Many of the tasks that the Unified Process defines involve using the UML and one or more models.

The Unified Process consists of cycles that may repeat over the long-term life of a system. A cycle consists of four phases: Inception, Elaboration, Construction and Transition. Each cycle is concluded with a release, there are also releases within a cycle. Let's briefly review the four phases in a cycle [7]:

- **Inception Phase** - During the inception phase the core idea is developed into a product vision. In this phase, we review and confirm our understanding of the core business drivers. We want to understand the business case for why the project should be attempted. The inception phase establishes the product feasibility and delimits the project scope.
- **Elaboration Phase** - During the elaboration phase the majority of the Use Cases are specified in detail and the system architecture is designed. This phase focuses on the "Do-Ability" of the project. We identify significant risks and prepare a schedule, staff and cost profile for the entire project.
- **Construction Phase** - During the construction phase the product is moved from the architectural baseline to a system complete enough to transition to the user community. The architectural baseline grows to become the completed system as the design is refined into code.
- **Transition Phase** - In the transition phase the goal is to ensure that the requirements have been met to the satisfaction of the stakeholders. This phase is often initiated with a beta release of the application. Other activities include site preparation, manual completion, and defect identification and correction. The

transition phase ends with a postmortem devoted to learning and recording lessons for future cycles.

For each of the phases, there are a number of iterations of “mini-projects”. Each of the iterations normally goes through the same workflow, and that is as described below.

- **Requirements** - The primary activities of the Requirements workflow are aimed at building the use case model, which captures the functional requirements of the system being defined. This model helps the project stakeholders reach agreement on the capabilities of the system and the conditions to which it must conform. The use case model also serves as the foundation for all other development work.
- **Analysis** - The primary activities of the Analysis workflow are aimed at building the analysis model, which helps the developers refine and structure the functional requirements captured within the use case model. This model contains realizations of use cases that lend themselves to design and implementation works better than the use cases.
- **Design** - The primary activities of the Design workflow are aimed at building the design model, which describes the physical realizations of the use cases from the use case model, and also the contents of the analysis model. The design model serves as an abstraction of the implementation model.

The Design workflow also focuses on the deployment model, which defines the physical organization of the system in terms of computational nodes.

- **Implementation** - The primary activities of the Implementation workflow are aimed at building the implementation model, which describes how the elements of the design model are packaged into software components, such as source code files, dynamic link libraries (DLLs), and EJBs (Enterprise Java Beans).
- **Test** - The primary activities of the Test workflow are aimed at building the test model, which describes how integration and system tests will exercise executable components from the implementation model. The test model also describes how the team will perform those tests as well as unit tests. The test model contains test cases that are often derived directly from use cases. Testers perform black-box testing using the original use case text, and white-box testing of the realizations of those use cases, as specified within the analysis model. The test model also contains the results of all levels of testing.

As stated, the 5 workflows shown above cuts through the 4 phases and these workflows are iterated at each phase to achieve a certain major milestone of the phase.

Architecture of the Unified Process

The process has two structures or “two dimensions”, as some would like to call it:

- The horizontal axis represents time and shows the lifecycle aspects of the process as it unfolds.

- The vertical axis represents core process disciplines, which group activities logically by nature.

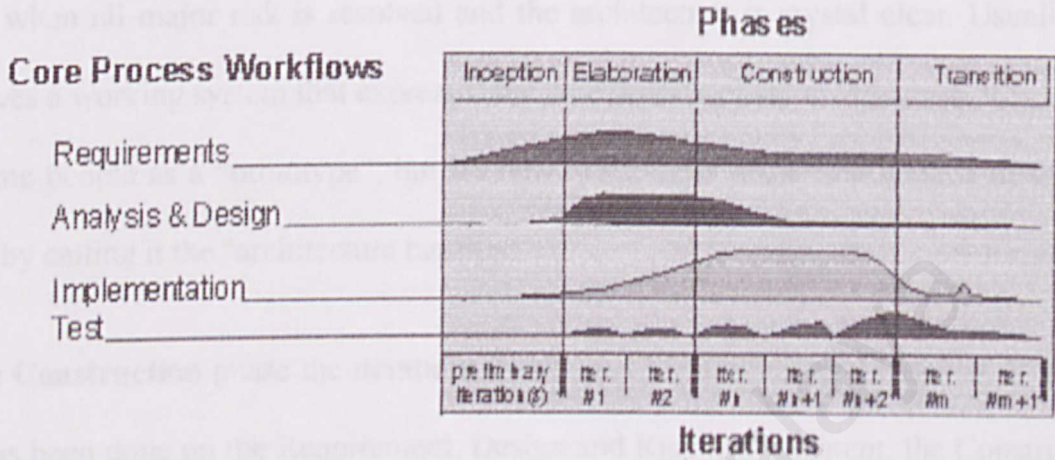


Figure 3.1: The Architecture of the Unified Process

As can be seen, the project starts at the beginning of the **Inception** phase. Here Management is heavily involved in determining Requirements to flesh out just what is needed. At first these are high level, objective oriented requirements. Specific ones come later in the second iteration of the Inception phase. For example the first Inception iteration may have the goal of "Determine high level requirements and probable feasibility". The second iteration's objective might be "Determine feasibility, final concept, and get all stakeholder agreement."

The **Elaboration** phase involves much more work. A "Go" decision has been made. Emphasis shifts to somewhat finalizing Requirements and doing high level Design, a.k.a architecture. Since architecture involves key decision, finalizing the architecture also

resolves most risk. Other risks, such as staff availability, market window timing, quality levels, etc, are met in a variety of ways as the project elaborates on the concept produced by the Inception phase. Interactions are used to compose an architecture the product can be built on, while addressing wave after wave of risk. The phase and last iteration are done when all major risk is resolved and the architecture is crystal clear. Usually this involves a working system that expresses the core aspects of the architecture. It is known to some people as a “prototype”, but it's best to consider it the first version of the real thing by calling it the “architecture baseline”.

In the **Construction** phase the iterations build upon the architectural baseline. If a good job has been done on the Requirement, Design and Risk Management, the Construction phase should go smoothly. There are a variety of ways to organize the workflows here. A lot of resources are being channeled here, simply because it is the “core” of the project itself. The phase ends when the product's coding is complete or further final work is best done in the customer's environment, such as when legacy data, embedded work or complex system changeover is present.

The **Transition** phase focuses transition to the customer's control. Actual transition may involve packaging, old system migration, hardware integration, training, or other domain specific transition work. It ends when the customer is satisfied they can begin using the product well. At this point the project is over.

3.2.2.2 The Three Characteristics of the Unified Process [8]

The three characteristics of the Unified Process have already been stated back in the second chapter. However, in this section of the third chapter an elaboration upon the three characteristics can be viewed below.

Use Case Driven

A use case is a sequence of actions, performed by one or more actors (people or non-human entities outside of the system) and by the system itself, that produces one or more results of value to one or more of the actors. One of the key aspects of the Unified Process is its use of use cases as a driving force for development. The phrase *use case driven* refers to the fact that the project team uses the use cases to drive all development work, from initial gathering and negotiation of requirements through code. Use cases are highly suitable for capturing requirements and for driving analysis, design, and implementation for several reasons.

- Use cases are expressed from the perspective of the system's users, which translates into a higher comfort level for customers, as they can see themselves reflected in the use case text. It's relatively difficult for a customer to see himself or herself in the context of requirements text.
- Use cases are expressed in natural language (English or the native language of the customers). Well-written use cases are also intuitively obvious to the reader.
- Use cases offer a considerably greater ability for everyone to understand the real requirements on the system than typical requirements documents, which tend to contain a lot of ambiguous, redundant, and contradictory text. Ideally, the

stakeholders should regard use cases as binding contracts between customers and developers, with all parties agreeing on the system that will be built.

- Use cases offer the ability to achieve a high degree of traceability of requirements into the models that result from ongoing development. By keeping the use cases close by at all times, the development team is always in touch with the customers' requirements.
- Use cases offer a simple way to decompose the requirements into chunks that allow for allocation of work to sub teams and also facilitate project management.

Architecture-Centric

It is the fundamental organization of the system as a whole. Aspects of an architecture include static elements, dynamic elements, how those elements work together, and the overall architectural style that guides the organization of the system. Architecture also addresses issues such as performance, scalability, reuse, and economic and technological constraints.

The Unified Process specifies that the architecture of the system being built, as the fundamental foundation on which that system will rest, must sit at the heart of the project team's efforts to shape the system, and also that architecture, in conjunction with the use cases, must drive the exploration of all aspects of the system. One might think of the architecture as expressing the common vision that all members of the team must share in order for the resulting system to be suitably robust, flexible, expandable, and cost-effective.

In the context of the process, architecture is primarily specified in terms of views of six models. These views reflect the "architecturally significant" elements of those models; taken together, the views form the architecture description. The project team initializes the architecture description early, then expands and refines it during virtually all the activities of the project.

The following subsections discuss the key reasons why architecture is so important to the Unified Process.

- **Understanding the Big Picture**

The tools and techniques available to developers for building software are growing increasingly powerful. For better or worse, though, software itself, especially with its new focus on distributed computing, is getting considerably more complex as well, and there aren't any indications that the tools and techniques will "catch up" any time soon. Also, customers' attention spans are becoming shorter and shorter as their demands on development teams grow more sophisticated. The result is that it's very difficult for all but a few especially gifted people to understand—really understand—most software systems to any meaningful extent. The architecture description is meant, first and foremost, to facilitate an understanding of the architecture of the system being built. Rigorous modeling, and careful attention to the readability of the associated UML diagrams and supporting text, will go a long way toward turning the architecture description into the fulcrum for increased understanding of the "big picture" of the new system.

- **Organizing the Development Effort**

A sound architecture explicitly defines discrete chunks of the system, as well as the interfaces among the various parts of the system. It also makes effective use of one or more architectural patterns, which help shape the development effort on various levels. (Client/server, three-tier, and N-tier are all examples of well-known architectural patterns. Other patterns focus on things like object request brokers [ORBs], which sit at the center of systems that use distributed components, and virtual machines, such as the one on top of which Java runs.) By using this aspect of architecture effectively, the project team can increase the chances that communication across sub teams will add value to the effort.

- **Facilitating the Possibilities for Reuse**

Well-constructed software architecture offers solid "scaffolding" on which components can reside and work gracefully with each other, while making it easy for teams building other systems to identify opportunities for possible reuse of any or all of those components. The bottom line is that the less time a team has to spend focusing on building new components, the more time it can spend on understanding the customers' problems and modeling the solutions.

- **Evolving the System**

Maintaining and enhancing a system tends to occupy more time over the life of that system than it took to build it in the first place. When development projects find

themselves operating in mythical "Internet time," with technologies evolving faster and business models changing more frequently than ever before, there's no question that a system of any size and complexity will be subject to evolutionary changes of a healthy magnitude. Having a solid architecture in place offers a set of essential reference points on which future development work can rely. An architecture that's been built such that changes in one part of the system almost never have adverse effects on other parts of the system also greatly enhances team members' ability to evolve the system effectively and efficiently.

- **Guiding the Use Cases**

In one sense, use cases drive the architecture of a software system, since the use cases do drive all of the development effort. In another sense, however, the architecture guides the selection and exploration of use cases. Decisions that architects must make about things like middleware, system software, legacy systems, and so forth, have a strong influence on the choice of which use cases the team focuses on at what point in the project. The basic idea, then, is to focus on those use cases that will add value to the architecture, which in turn helps shape the content of those use cases and the nature of the work involved in developing the system from them.

Iterative and Incremental

The third characteristic of the Unified Process is its iterative and incremental nature. An iteration is a mini-project that results in a version of the system that will be released

internally or externally. This version is supposed to offer incremental improvement over the previous version, which is why the result of an iteration is called an increment.

The following subsections describe the advantages of iterative and incremental development.

- **Logical Progress toward a Robust Architecture**

An earlier section, "Architecture-Centric," describes the central place of architecture in the Unified Process. The process specifies how the project team should focus on particular aspects of the architecture during each of the iterations of the system. During early iterations, the team puts together a candidate architecture that offers the beginnings of a solid foundation; later iterations find the team expanding the vision of the full architecture, which in turn influences most, and in some cases all, of the development tasks being performed as part of a given iteration. Building the architecture in an iterative and incremental fashion enables the team to make necessary major changes early in the process at considerably less cost than they would inflict later in the project.

- **Dealing With Ongoing Changes in Requirements**

The Unified Process advocates breaking the system down into builds, where each build is a working version of some meaningful chunk of the full system. By focusing on bounded sets of use cases and making effective use of prototypes, the project team and the customers can negotiate requirements on an ongoing basis, thus

reducing the (often very large) risk associated with trying to specify all of the requirements up front. One of the reviewers of the manuscript for this book indicated that his company practices "ruthless prioritization," which involves dealing with changing requirements by aggressively identifying priorities and eliminating lower-priority features from consideration.

- **Greater Flexibility to Change the Plan**

As risks become greater, as delays occur, and as the environment becomes more unstable, the team is able to make necessary adjustments on a relatively small scale and propagate those adjustments across the entire project. The goal is to isolate problems within iterations and deal with them on a relatively small scale, rather than allowing them to spread.

- **Continuous Integration**

By continually integrating new increments, the development team is also able to isolate problems that it might bring to the system and address those problems in ways that don't disrupt the integrity of the working system. This kind of setup makes it easier for the team to go as far as throwing a particular increment away and starting over, since the process gives it the ability to define iterations that take less time to perform.

- **Early Understanding**

Well-defined iterations allow room to experiment and make mistakes, because those mistakes will be isolated such that their impact on schedule and budget can be minimized. As work proceeds, the team can leverage its understanding of what it's trying to build and the associated risks, thus building momentum, which in turn enables the team to make continuous improvements in the way it goes about its tasks.

- **Ongoing Focus on Risk**

Perhaps the most important advantage that iterative and incremental development, as defined by the Unified Process, brings to the table is the project team's ability to focus its efforts on addressing the most critical risks early in the life cycle. The team has a mandate to organize iterations based on addressing risks on an ongoing basis; the goal is to mitigate risks to the greatest extent possible during each iteration, so each iteration poses fewer risks of less importance than its predecessors.

3.2.2.3 The Primary Models of the Unified Process

There are 6 primary models to that of the Unified Process. Below is the diagram to allow better understanding of the concept:

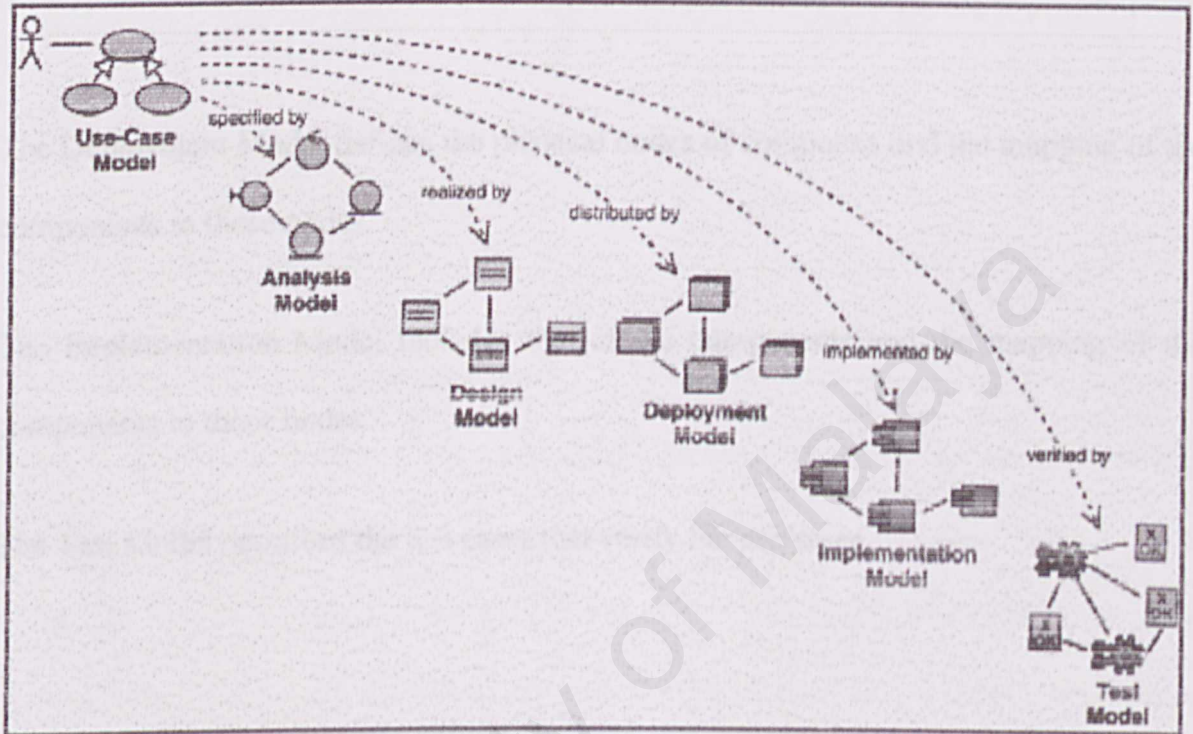


Figure 3.2: The Primary Models of the Unified Process.

The diagram above and the models in the diagrams are described briefly below:

The Use Case Model interacts with all the use cases and their relationships to users.

The Analysis Model has two purposes:

- To refine the use cases in more detail.
- To make an initial allocation of the behavior of the system to a set of objects that provides the behavior.

The Design Model defines:

- The static structure of the system as subsystems classes and interfaces.
- The use cases realized as collaborations among them.

The Deployment Model defines the physical nodes of computers and the mapping of the components to those nodes.

The Implementation Model includes that of the components and the mapping of the components to those nodes.

The Test Model describes the test cases that verify the use cases.

3.2.2.4 Justification of Methodology

The reason for choosing this methodology over other types of methodology is because that the Unified Process offers a lot more advantages than any other methodologies that has been researched for this particular project. Amongst the advantages are:

- It exists and is well documented, so there is no wheel to reinvent.
- It was introduced with a lot of fanfare and a tremendous organization behind it, like a well-orchestrated marketing campaign.
- It documents all the major artifacts that can be produced and used by a software development project from inception to testing of the final code.

- It has industry-wide acceptance of the artifacts of UML that it uses for documentation.
- It is rapidly becoming the "lingua franca" or the default methodology for software developers.
- It has several powerful tools that support artifact production.

3.3 Unified Modeling Language [9]

The Unified Modeling Language (UML) is a language for specifying, constructing, visualizing, and documenting the artifacts of a software-intensive system. First and foremost, the Unified Modeling Language fuses the concepts of Booch, OMT, and OOSE. The result is a single, common, and widely usable modeling language for users of these and other methods. Second, the Unified Modeling Language pushes the envelope of what can be done with existing methods. As an example, the UML authors targeted the modeling of concurrent, distributed systems to assure the UML adequately addresses these domains. Third, the Unified Modeling Language focuses on a standard modeling language, not a standard process.

The UML can model just about any types of application, running on any type and combination of hardware, operating system, programming language, and network. Built upon the MOF™ metamodel which defines class and operation as fundamental concepts, it's a natural fit for object-oriented languages and environments such as C++, Java, and the recent C#, but you can use it to model non-OO applications as well in, for example, Fortran, VB, or COBOL. *UML Profiles* (that is, subsets of UML tailored for

specific purposes) help you model Transactional, Real-time, and Fault-Tolerant systems in a natural way. Another one good thing about using the UML that is a must to be mentioned is that it allows for the user to be very much methodology-independent, for it provides for the user the ability to perform analysis and design, regardless of what kind of methodology the user uses.

Class diagrams are the backbone of almost every object-oriented system, including UML. They describe the static structure of a system.

3.3.1 Unified Modeling Language Diagrams

UML defines nine types of diagrams: class, object, use case, sequence, collaboration, statechart, activity, component, and deployment. These diagrams can be divided into two groups: Structural diagrams, which model the organization of the solution, and Behavioral diagrams, which model the functionality of the solution.

Structural Diagrams

Object Diagrams

- Class Diagram
- Object Diagram
- Component Diagram
- Deployment Diagram

Behavioral Diagrams

- Use Case Diagram
- Sequence Diagram
- Collaboration Diagram

- Statechart Diagram
- Activity Diagram

Below are the brief explanations regarding each of the diagrams:

Class Diagrams

Class diagrams are the backbone of almost every object oriented method, including UML. They describe the static structure of a system.

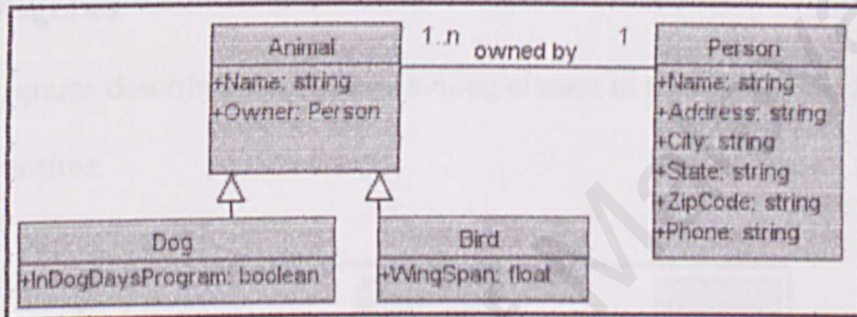


Figure 3.3: Class Diagrams

Object Diagrams

Object diagrams describe the static structure of a system at a particular time. They can be used to test class diagrams for accuracy.

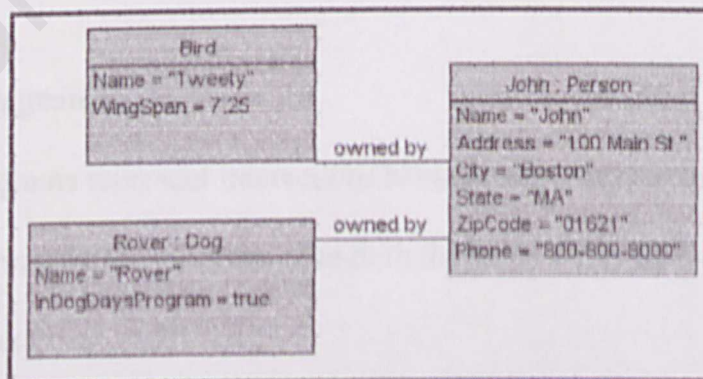


Figure 3.4: Object Diagrams

Use Case Diagrams

Use case diagrams model the functionality of system using actors and use cases.

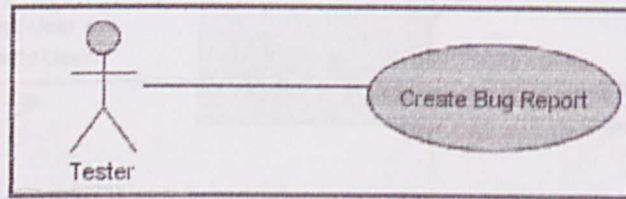


Figure 3.5: Use Case Diagrams

Sequence Diagrams

Sequence diagrams describes interactions among classes in terms of an exchange of messages over time.

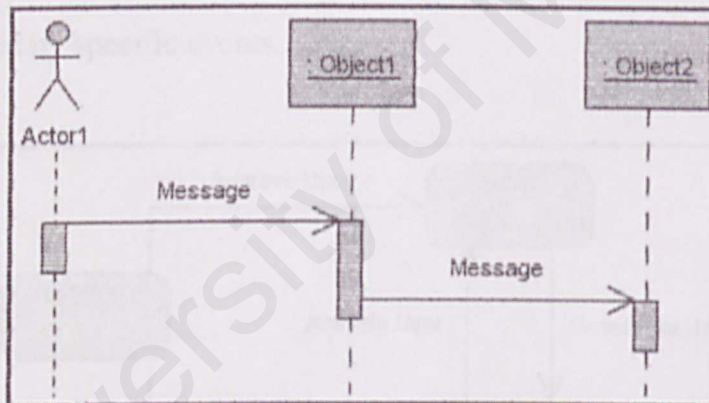


Figure 3.6: Sequence Diagrams

Collaboration Diagrams

Collaboration diagrams represent interactions between objects as a series of sequenced messages. Collaboration diagrams describe both the static structure and the dynamic behavior of a system.

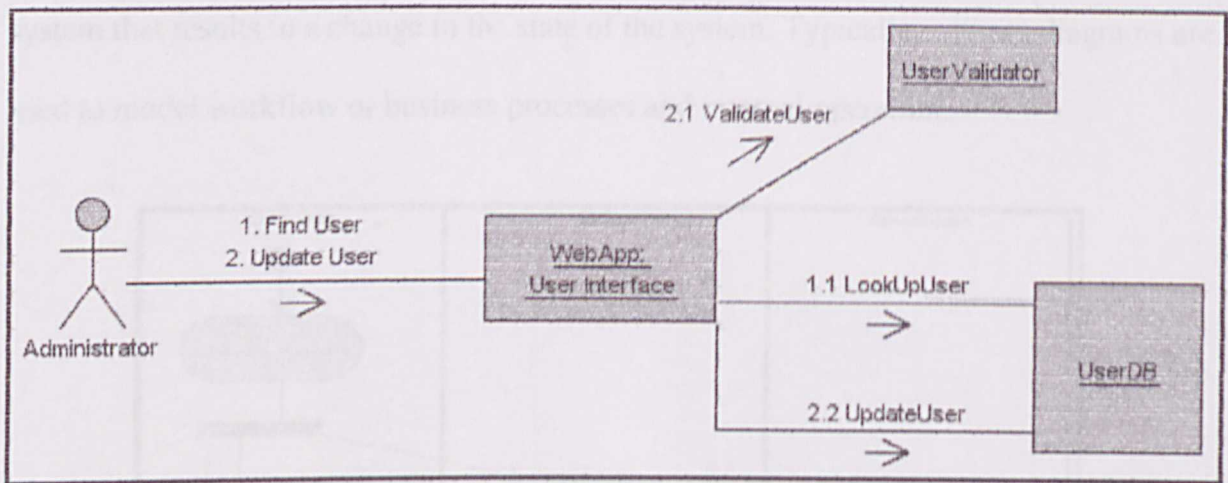


Figure 3.7: Collaboration Diagrams

Statechart Diagrams

Statechart diagrams describe the dynamic behavior of a system in response to external stimuli. Statechart diagrams are especially useful in modeling reactive objects whose states are triggered by specific events.

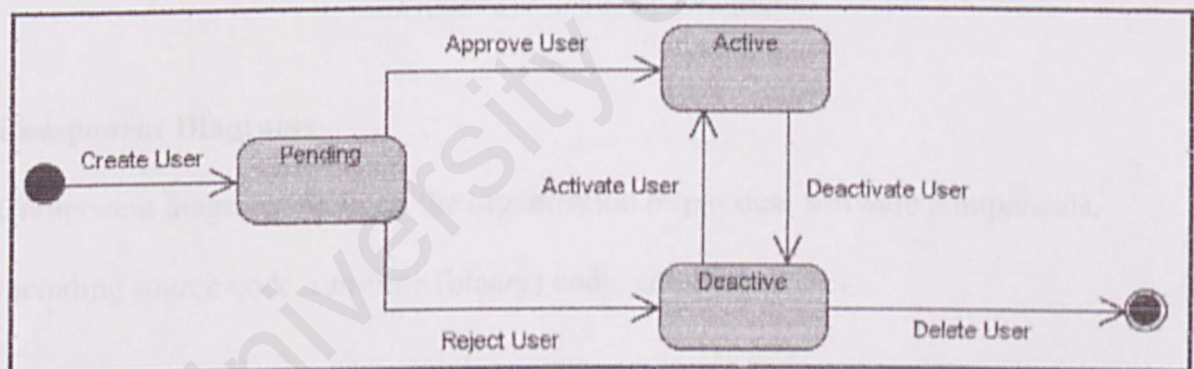


Figure 3.8: Statechart Diagrams

Activity Diagrams

Activity diagrams illustrates the dynamic nature of a system by modeling the flow of control from activity to activity. An activity represents an operation on some class in the

system that results in a change in the state of the system. Typically, activity diagrams are used to model workflow or business processes and internal operation.

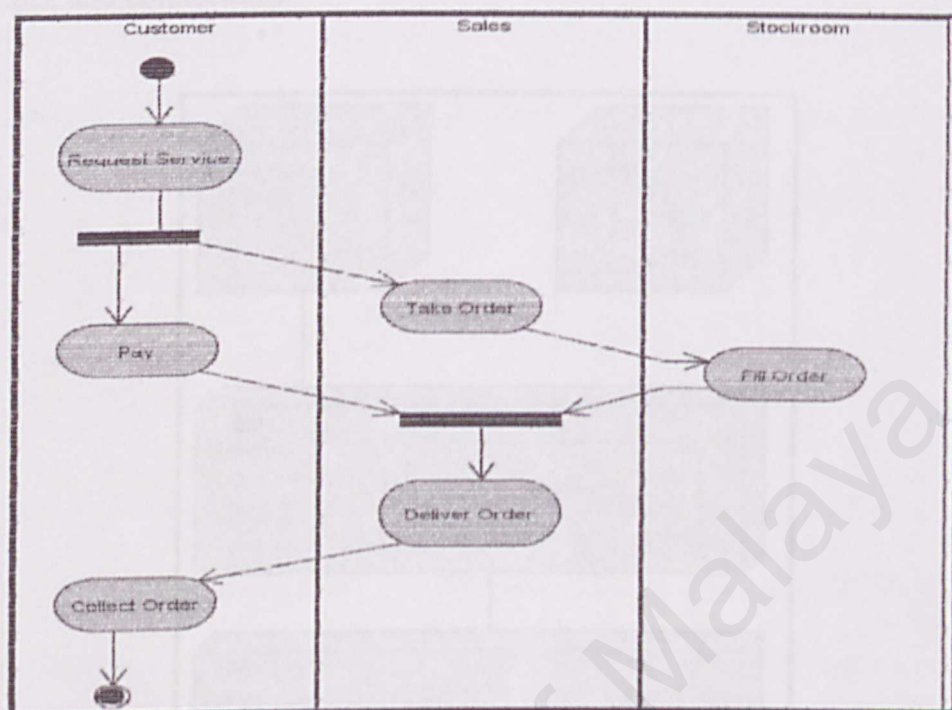


Figure 3.9: Activity Diagrams

Component Diagrams

Component diagrams describe the organization of physical software components, including source code, run-time (binary) code, and executables.

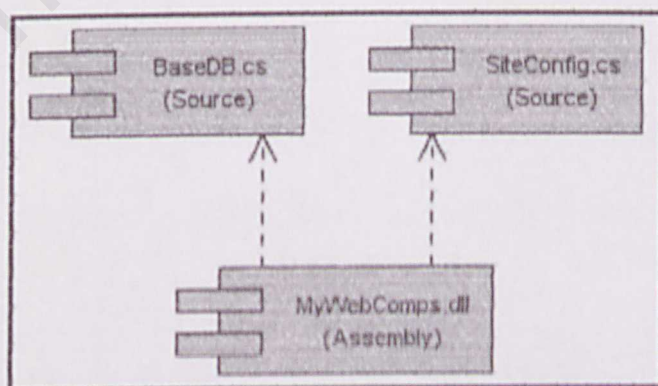


Figure 3.10: Component Diagrams

Deployment Diagrams

Deployment diagrams depict the physical resources in a system, including nodes, components, and connections.

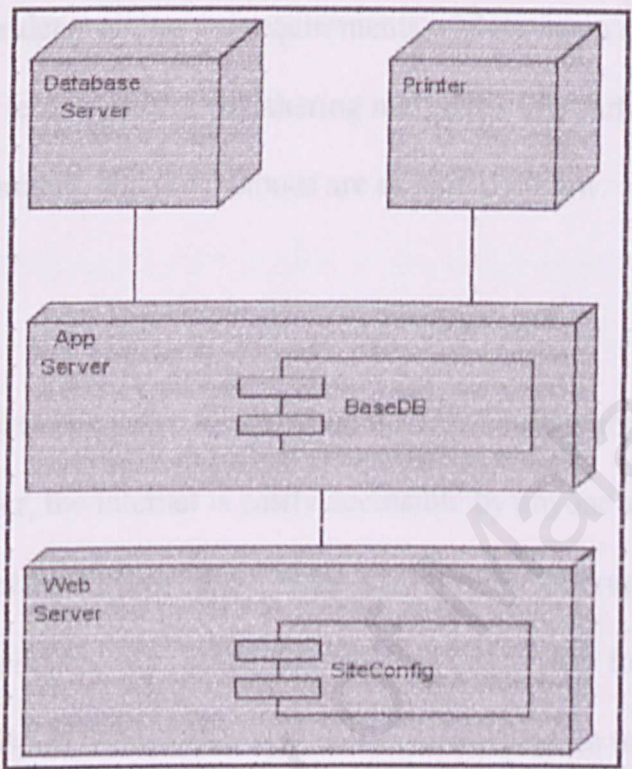


Figure 3.11: Deployment Diagrams

3.4 Information Gathering Methods

A proper and effective method for getting information is vital to ensure a sound understanding of the system in all aspects. The acquired information will serve as a good groundwork for determining the requirements of the system and also for designing the system. Thus, a few information gathering methods were performed in the midst of the research for the project, and the methods are as written below.

Internet Research

The internet is a powerful tool for looking up information regarding any subjects whatsoever. Moreover, the internet is easily accessible by anyone who can get a hold of a computer with an internet connection. More and more thesis projects, researches done by professors and scholars, are being put on the net in the forms of HTML file, document files and also PDF files. Thus, it comes as no surprise to say that most of the information gathered for this project comes from the information gotten from the internet itself.

Books and References

Books are also used while in the process of gathering information for the project. Books and references by scholars and respected authors are being used in this project, simply because they provide a more in depth explanation regarding certain information that is needed.

3.5 Conclusions on Tools and Technologies

Back in chapter two, few reviews regarding the types of operating system and programming languages were done, and in this section the programming language and operating system of choice shall be selected for the project.

Selected Operating System

The operating system of choice shall be that of the Windows XP Professional. It was chosen over that of the SuSE Linux because it provides a leaner learning curve than that of the latter. Moreover all this while people are more used to the Windows interface and environment, and with the few major upgrades and enhancement done to Windows XP Professional, one can never go wrong by choosing it as the operating system of choice.

Selected Programming Language

The programming language that has been chosen for this project is MATLAB 7.0 over that of Visual Basic. The reason is that MATLAB has an extensive library regarding that of image processing, and that alone is a good reason for choosing MATLAB. The extensive library also means that the user don't have to code the image processing process and coding from scratch and thus saves time in its construction phase. With all those advantages, MATLAB has indeed an edge over that of the Visual Basic programming language.

3.6 Chapter Summary

By choosing and adhering to a good software development process, this project aims to produce a system of commendable quality. The Unified Process methodology shall serve as a guideline as to what activities should be done at what stage. This will ensure a systematic way of developing the proposed system. Other than that, the information gathering methods played an important role indeed in finding the right information that is to be applied for the project. In this chapter also the operating system and programming language of choice were chosen for the project.

CHAPTER 4

SYSTEM ANALYSIS

4.1 Techniques for Requirements Elicitation

Requirements Elicitation is the process of discovering the requirements for a system by communication with customers, system users and others who have a stake in the system development. Other method of requirements elicitation is by doing research by getting information from sources, such as the internet, books and references.

As for this project, the first method that was used for eliciting the requirements for the face recognition system was by doing some research upon the existing face-based system that already existed. Most of the existing system's information was gathered from the internet. After that the advantages and disadvantages were being focused upon, and the advantages that can be used for this project shall be adopted and the disadvantages of the previous system shall be avoided.

The next method of eliciting requirements for the face recognition system was by using use cases. Use Cases, like storyboards, identifies the who, what, and how of system behavior. Use Cases also describes the interactions between a user and a system, focusing on what they system "does" for the user. The Use Case model describes the totality of the systems functional behavior. Once the Use Case model is done, then the requirements can be easily outlined.

4.2 System Requirement Analysis

System requirement sets out the system services and constraints. An analysis is carried out to determine the requirements of the system, which are normally divided into two categories which are the functional requirements and the non-functional requirements. Functional requirements are statement of services that the system should provide, how the system suppose to work and react towards or under certain conditions. Non-functional requirements are constraints on the services or functions offered by the systems. Below are listed the functional and the non-functional requirements for this particular project.

4.2.1 Functional Requirements

- **Load the set of images to be processed:** A set of images is being feed into the system as for to calculate the eigenvectors, which is a face space that is calculated for the set of face images.
- **Load a single image as input:** An image is being feed through the system.
- **Perform mathematical calculations:** Mathematical calculations are being done to calculate how close the input image is to the value of the eigenvectors.
- **Display image and recognize image:** After the mathematical calculations have been done, the image that the input image matches closest to shall be displayed.

4.2.2 Non-Functional Requirements

- **Reliability:** The system must be able to perform the processes that it is suppose to be able to do, and come up with a respectable and acceptable recognition rates that is dependable.
- **Ease of use:** The system must be easy to be used by the user, as for user would be let down by a system that is hard to understand or have a slightly steeper learning curve to it.
- **Accuracy:** The system must be able to produce the results as accurate as possible as the expected outcome.
- **Maintainability:** The system must be easy to maintain in order for it to be usable in future times.

4.3 Proposed Algorithm

The proposed algorithm, as was mentioned since the first chapter of the project, is that of the eigenface algorithm. The eigenface approach is actually a principal component analysis method, in which a small set of characteristic pictures are used to show the variation between face images. These are actually the eigenvectors (eigenfaces) of the covariance matrix for the set of face images. The eigenvectors are then used to define the subspace of face images that is also known as face space. As for the recognition part, a new image is projected into the subspace spanned by the eigenvectors

(eigenfaces) and then classifying them by comparing its position in the face space with the positions of known individuals. If the face image resembles as close as possible to the images of the known individuals, then it will be classified as a “known” face image, and it will be classified as “unknown” face image if otherwise.

4.3.1 Justification of the Chosen Technique

The eigenface technique was chosen because of a few reasons that need to be mentioned here. First and foremost is that amongst the other techniques available, this is the easiest algorithm to be applied to that of a face recognition system. The eigenface method also is being used widely and thus it is easier to get references regarding the methods and the coding and also the algorithm itself. Last but not least, since this is an undergraduate project, and the first face recognition project by the author, it is best to go with the most well known technique around, and with that, the eigenface method is, without a doubt, the more reasonable technique to start of with.

4.3.2 Overview of the Proposed Algorithm

The proposed algorithm is that of the eigenface algorithm, which is easy to apply and also to maintain when compared to other algorithm. The flow of the system shall be shown in the diagram below.

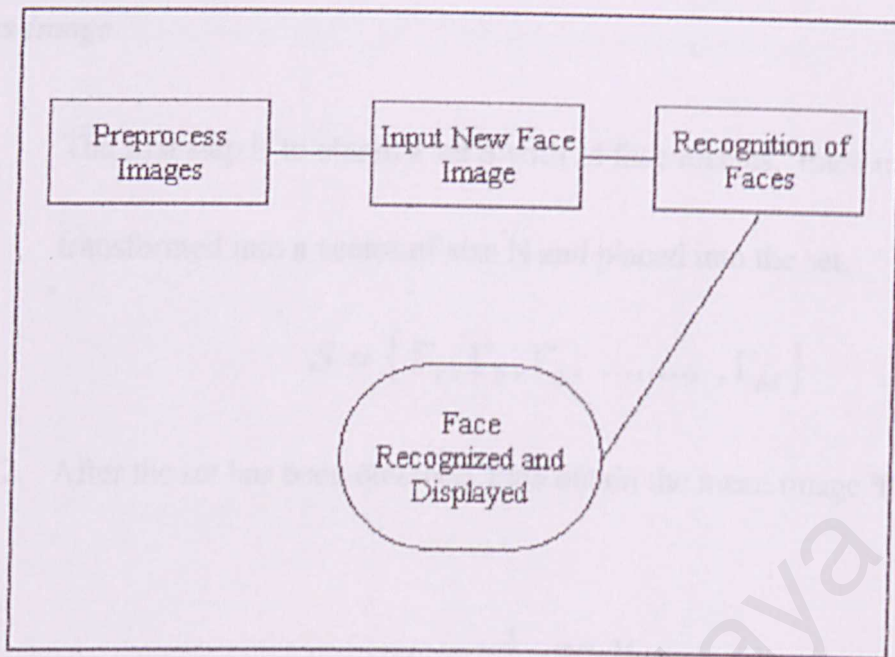


Figure 4.1: The Face Recognition System Work Flow

As can be seen from the above diagrams, the system works in such a way firstly it processes the face images, producing the eigenfaces. After the processing has been done, a new face image is being fed into the system. After that, the new face image is then being matched up against the eigenfaces and determined whether it resembles any of the faces in the system or not. Finally, after much calculation, the system shows the result, and that is if the face matches the eigenfaces, it will be displayed, and if it doesn't, then it will be shown as an unknown face.

Basically the eigenface mathematical algorithms are about the same from one to another. As for that, below the author shall show that of the mathematical and theoretical aspect of the face recognition system as to give the user a better understanding of it.

Preprocess Image

1. The first step is to obtain a set S with M face images. Each image is transformed into a vector of size N and placed into the set.

$$S = \{ \Gamma_1, \Gamma_2, \Gamma_3, \dots, \Gamma_M \}$$

2. After the set has been obtained, then obtain the mean image Ψ

$$\Psi = \frac{1}{M} \sum_{n=1}^M \Gamma_n$$

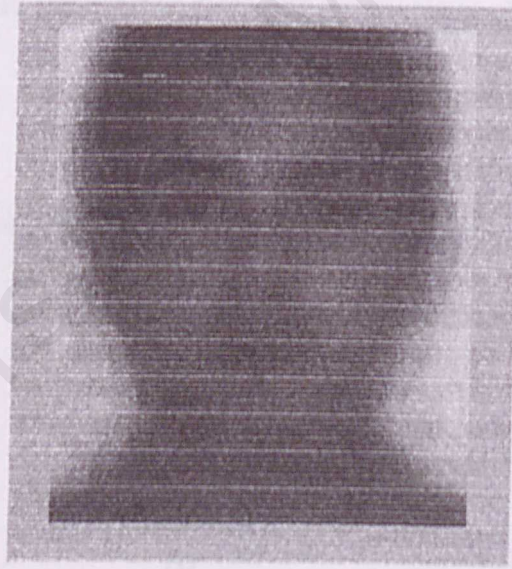


Figure 4.2: The mean of an image.

3. Then the difference Φ between the input image and the mean image will have to be done.

$$\Phi_i = \Gamma_i - \Psi$$

4. Next thing is to seek a set of M orthonormal vectors, \mathbf{u}_n , which best describes the distribution of the data. The k^{th} vector, \mathbf{u}_k , is chosen such that

$$\lambda_k = \frac{1}{M} \sum_{n=1}^M (\mathbf{u}_k^T \Phi_n)^2$$

is a maximum, subject to

$$\mathbf{u}_l^T \mathbf{u}_k = \delta_{lk} = \begin{cases} 1 & \text{if } l = k \\ 0 & \text{otherwise} \end{cases}$$

Note: \mathbf{u}_k and λ_k are the eigenvectors and eigenvalues of the covariance matrix \mathbf{C}

5. After that the covariance matrix \mathbf{C} shall be obtained in the following manner

$$\begin{aligned} \mathbf{C} &= \frac{1}{M} \sum_{n=1}^M \Phi_n \Phi_n^T \\ &= \mathbf{A} \mathbf{A}^T \end{aligned}$$

$$\mathbf{A} = \{ \Phi_1, \Phi_2, \Phi_3, \dots, \Phi_n \}$$

6. \mathbf{A}^T

$$L_{mn} = \Phi_m^T \Phi_n$$

7. Once we have found the eigenvectors, v_l, u_l

$$u_l = \sum_{k=1}^M v_{lk} \Phi_k \quad l = 1, \dots, M$$



Figure 4.3: These are examples of the eigenfaces set of images

Input New Face Image

The next thing would be to input single face image to the face recognition system. The face image must be that of the same size as that of the preprocessed images (e.g. 24 x24 pixels) that has been turned into eigenfaces. The face image then will be mapped to the eigenfaces and the recognition phase will follow.

Recognition of Face

1. A new face is transformed into its eigenface components. Firstly compare the input image with the mean image and multiply their difference with each eigenvector of the L matrix. Each value would represent a weight and would be saved on a vector Ω .

$$\omega_k = u_k^T (\Gamma - \Psi) \quad \Omega^T = [\omega_1, \omega_2, \dots, \omega_M]$$

2. After that is the selection of which face class provides the best description for the input image. This is done by minimizing the Euclidean distance

$$\varepsilon_k = \|\Omega - \Omega_k\|^2$$

3. The input face is considered to belong to a class if ε_k is below an established threshold θ_b . Then the face image is considered to be a known face. If the difference is above the given threshold, but below a second threshold, the image can be determined as a unknown face. If the input image is above these two thresholds, the image is determined NOT to be a face.

Face Recognized and Displayed

The face image that has been transform into eigenface and then mapped to the set of eigenfaces and recognized by the system as a known face image, shall be displayed.

4.4 Hardware Requirements

In order for the face recognition system to run smoothly, a set of hardware specifications were made, and these hardwares are going to be used in the development process of the system. Thus, the hardware specification also serves as to show the user what kind of hardwares works best for the system. The hardwares are as shown below.

Hardware	Specification
Processor	AMD Athlon XP 2200/ Intel Pentium 1,2,3,4
Memory	256 MB RAM (SD/ DDR/ RDRAM)
CD ROM	52X CD ROM
Hard Drive	40 GB MAXTOR 7200 RPM
Keyboard	PS2 / USB Keyboard (input)
Mouse	PS2 / USB Mouse Optical (input)

Table 4.1: Hardware Specification.

4.5 Software Requirements

Software that was used to build the face recognition system is divided into two categories, which are:

- 1. Software that act as a platform for the system
- 2. Software used to code the system

For the first category, Windows XP Professional shall be used and are mostly used throughout the documentation phase up to the developments and testing phase. For the second category, Mathworks MATLAB 7.0, SmartDraw Professional, and Microsoft Words shall be used throughout the two semesters.

4.6 Face Database

The face database consists of 60 face images of 18 people altogether. The face database is obtained from the Sluggish Software website [10], which was used for their face recognition system, and the face database consist also of 9 females and 9 males, each are about 20-35 years of age. Since the face database would take up the whole page, the face database is being placed in the Appendix section at the end of the report.

4.7 Chapter Summary

This chapter mainly deals with the requirements for the system. At the beginning, the technique used to elicit the system requirements was reviewed. After that a list of functional and non-functional requirements were specified. Other than that, the hardware and software specifications were also mentioned in this chapter.

One of the main components of the system, and that is the algorithm itself, was proposed and explained as to give a clearer picture of how the system works.

CHAPTER 5

SYSTEM DESIGN

In the previous chapter we have outlined the analysis of the system itself, which are the techniques for requirements elicitation and the functional and non-functional requirements. Also in the previous chapter we have discussed about the proposed algorithm, which is the eigenface algorithm, a well known and widely used algorithm indeed. Later in the previous chapter, the hardware and the software requirements were also mentioned. Altogether, the previous chapter can be seen as an indicator of “WHAT NEEDS TO BE DONE”. In this chapter however, we shall discuss about “HOW IT WILL BE DONE”. Taking the analysis from the previous chapter, we shall use it to design the system as a whole, starting from the modeling of the system straight up to the design of the interface itself.

5.1 Design of the System Function

Everything regarding the system is encompassed within a single module. It deals with all the aspect of the system, which are the face recognition and the pre-processing of images. It uses the eigenface algorithm to recognize faces.

As was stated before in the previous chapters, the system works by obtaining a set of face images for it to do some pre-processing on the images. The result of the pre-processing is that the system comes up with an eigenvectors (or eigenfaces), which is to

be used as a template for recognizing face images. After that the user needs to feed in a single face image into the system, be it a known or an unknown face image. The system will then change the face image into an eigenface and then map the eigenface to the face space, which is actually a set of engenfaces of the previous pre-processing of the set of face images. If the eigenface of the face image that has been fed into the system resembles closely to any of the eigenfaces of the set of images, the system will then display the image that the face image resembles closely to. It otherwise, the system will then label the face image as an unknown face image.

From the system, two functional requirements were the pre-processing of face images and the face recognition process. There are other functional requirements in the system, but both of these shall be used as for to model the process using that of the sequence diagrams. The diagrams are shown on the next page.

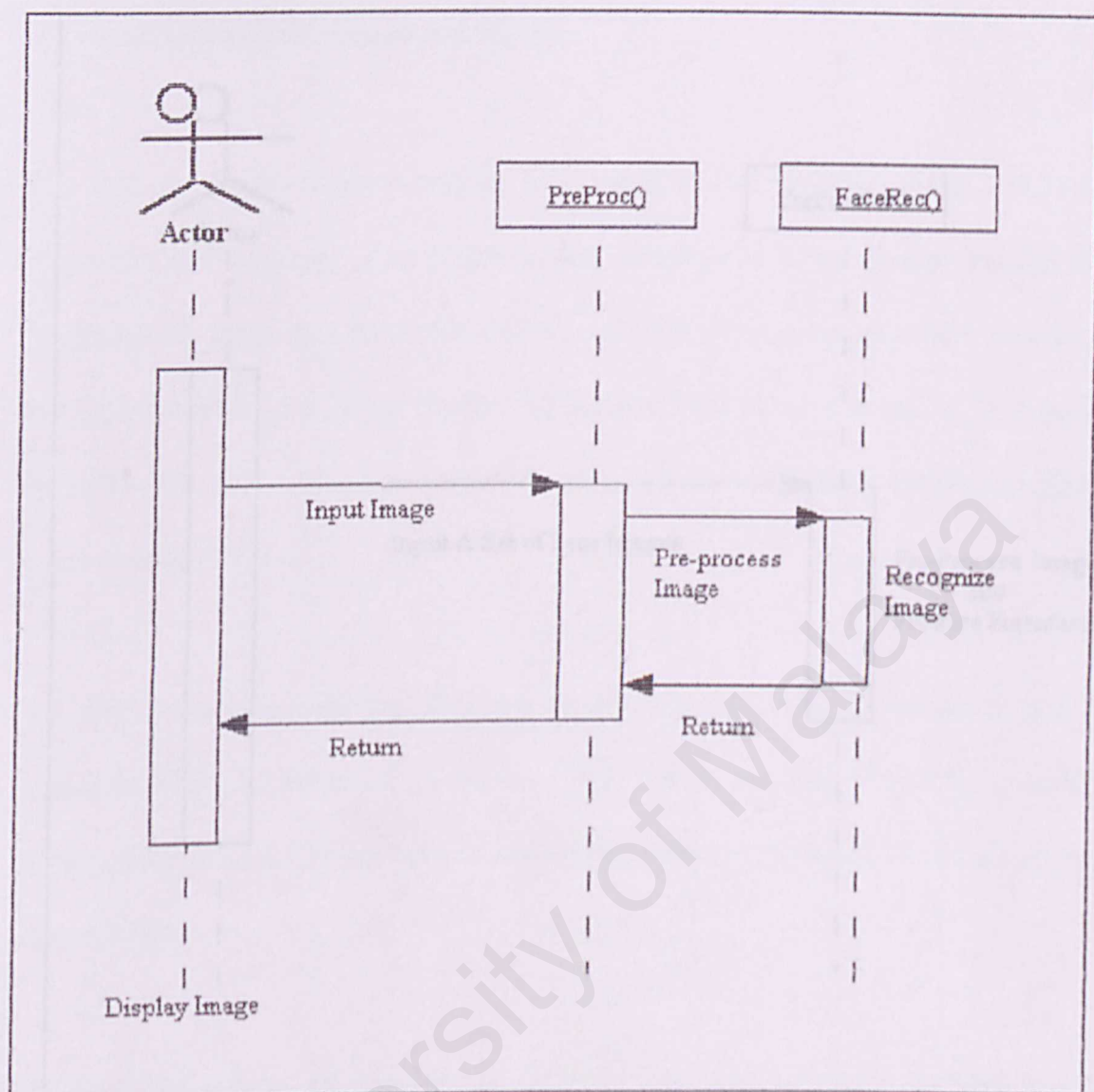


Figure 5.1: Sequence Diagram for Face Recognition Process

The actor (the user) inputs a single face image into the system. The system then passes the face image to the PreProc object, which does the preprocessing of the face image. The pre-processing results in the face image being turned into a single eigenface. The eigenface, will then be passed to the FaceRec() object, which matches the single eigenface to the eigenfaces of obtained from the setoff face images. If it is being recognized as a known face, then the system will display the face image that matches it.

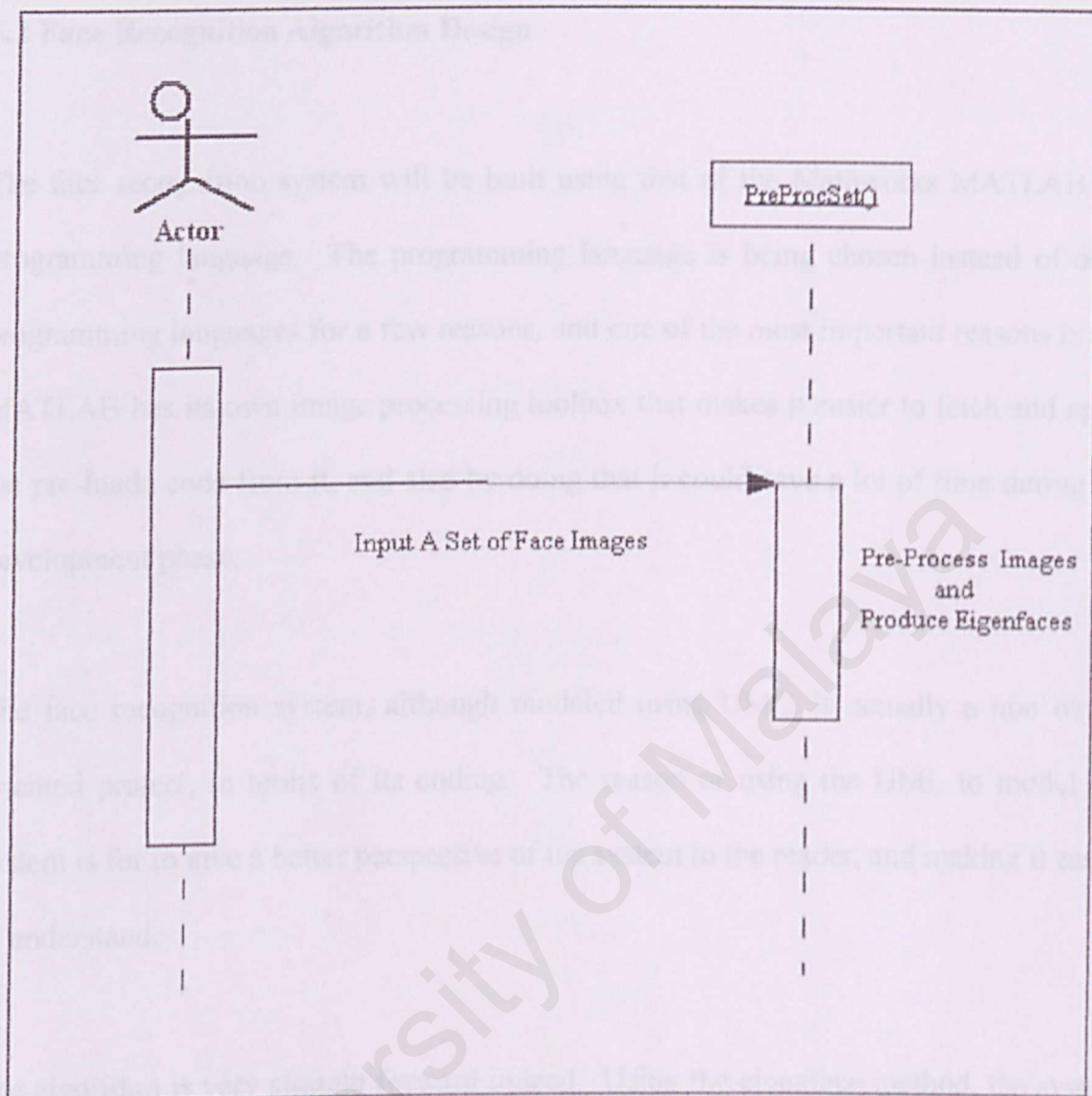


Figure 5.2: Sequence Diagram for Pre-Processing of Face Images Process

This sequence diagram shows that of the pre-processing of the set of face image, selected to become the eigenfaces that will be used or recognition of other face images that will be fed into the system. The actor feed into the system a set of face images. It will then be pre-processed into eigenfaces, and that task is being done by the PreProSet() object. The eigenfaces will then be used as a base to recognize every other face images that shall be fed into the system.

5.2 Face Recognition Algorithm Design

The face recognition system will be built using that of the Mathworks MATLAB 7.0 programming language. The programming language is being chosen instead of other programming languages for a few reasons, and one of the most important reasons is that MATLAB has its own image processing toolbox that makes it easier to fetch and apply the pre-made code from it, and also by doing that it could save a lot of time during the development phase.

The face recognition system, although modeled using UML, is actually a non object oriented project, in terms of its coding. The reason or using the UML to model the system is for to give a better perspective of the system to the reader, and making it easier to understand.

The algorithm is very straight forward indeed. Using the eigenface method, the system first must pre-process a set of face images, turning them into eigenfaces. After that a single face image is fed through the system and then pre-processed to become that of an eigenface. Then the eigenface of the single face image will be mapped to the eigenfaces of the set of face images, and the similarities between the two are calculated. If it resembles closely to a certain eigenface, then the picture of the person shall be displayed. If not, the face shall be identified as an unknown face. To have a better understanding of the system, an activity diagram is provided on the next page let the readers have a picture of how it works.

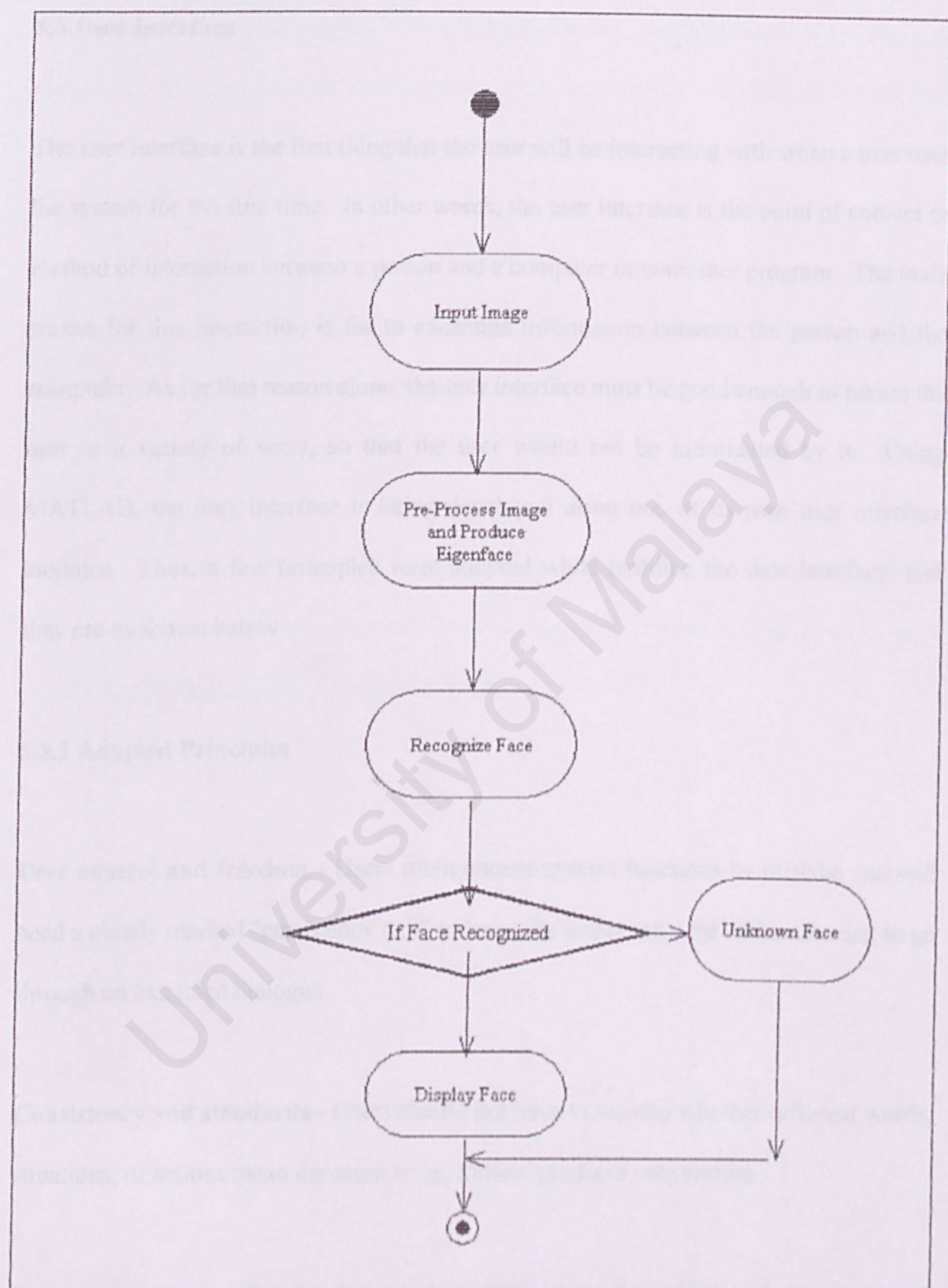


Figure 5.3: Activity Diagram for Face Recognition Process

5.3 User Interface

The user interface is the first thing that the user will be interacting with when a user uses the system for the first time. In other words, the user interface is the point of contact or method of interaction between a person and a computer or computer program. The main reason for this interaction is for to exchange information between the person and the computer. As for that reason alone, the user interface must be good enough to please the user in a variety of ways, so that the user would not be intimidated by it. Using MATLAB, the user interface is being developed using one of its own user interface modules. Thus, a few principles were adopted while building the user interface, and they are as shown below.

5.3.1 Adopted Principles

User control and freedom - Users often choose system functions by mistake and will need a clearly marked "emergency exit" to leave the unwanted state without having to go through an extended dialogue.

Consistency and standards - Users should not have to wonder whether different words, situations, or actions mean the same thing. Follow platform conventions.

Error prevention – Careful design which helps in cutting away with the unwanted errors.

Recognition rather than recall - Make objects, actions, and options visible. The user should not have to remember information from one part of the dialogue to another. Instructions for use of the system should be visible or easily retrievable whenever appropriate.

Flexibility and efficiency of use - Accelerators -- unseen by the novice user -- may often speed up the interaction for the expert user such that the system can cater to both inexperienced and experienced users.

Aesthetic and minimalist design - Dialogues should not contain information which is irrelevant or rarely needed. In other words it should be as simple looking as possible and also easily maneuvered through the system.

5.3.2 User Interface for the Face Recognition System

Below is the proposed user interface for the Face Recognition system.

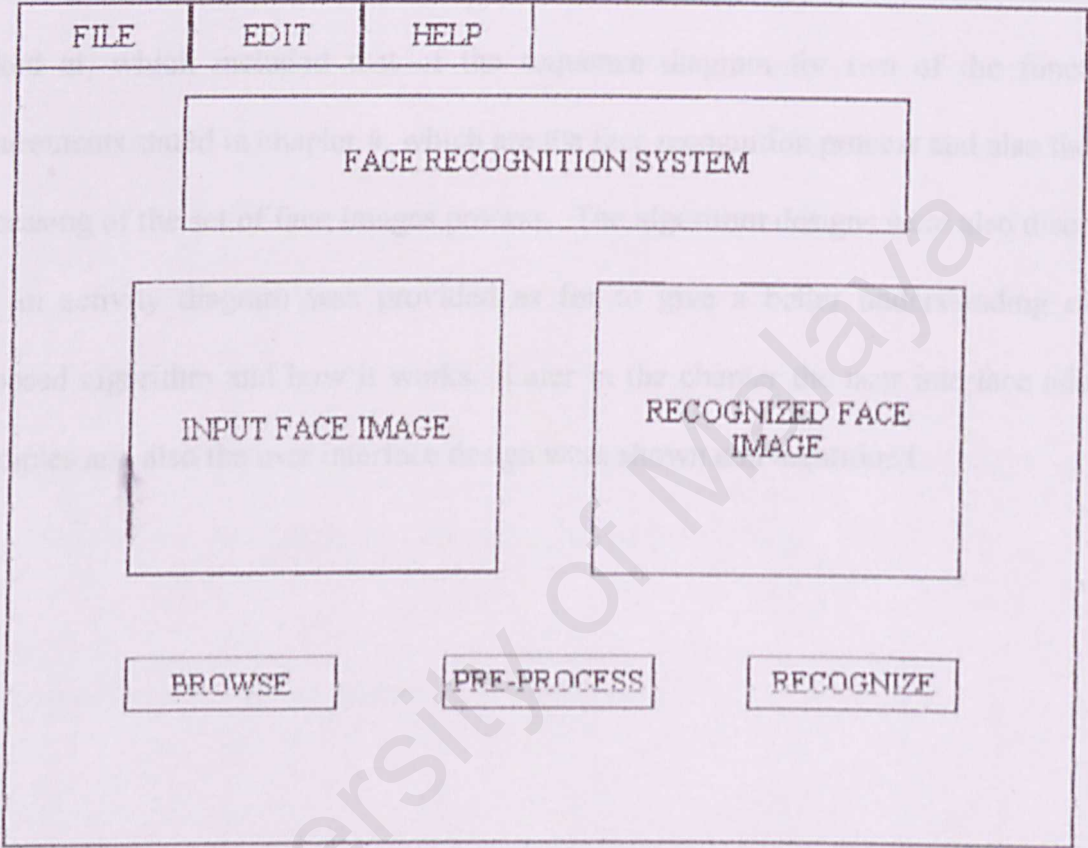


Figure 5.4: User Interface for the Face Recognition system.

5.4 Chapter Summary

In this chapter the overall architecture of the system and the user interface were being discussed to a certain extent. First off the design of the system functions were being looked at, which included that of the sequence diagram for two of the functional requirements stated in chapter 4, which are the face recognition process and also the pre-processing of the set of face images process. The algorithm designs were also discussed and an activity diagram was provided as for to give a better understanding of the proposed algorithm and how it works. Later in the chapter the user interface adopted principles and also the user interface design were shown and mentioned.

6.1 Processing of the Input Image

Any images are actually made up of pixels, the very basics of an image in an image. Therefore, any image processing task requires the system to be able to extract the properties of each and every pixel in order to operate on it. In Microsoft Visual Basic 6.0, the image is every pixel of the image is being viewed as a form of a data array where the value for each pixel is stored in it. The 2 main functions that are being used to extract and use the pixels are as shown below:

CHAPTER 6

SYSTEM IMPLEMENTATION

In this chapter, the implementation technique of the system shall be discussed at length. The main thing regarding the system at hand is that the programming language that has been picked to build the system has been changed from MATLAB to that of Microsoft Visual Basic 6.0. As can be seen this is indeed different from what was proposed beforehand. The reasons for it shall be discussed in later segments of this chapter, as for now it is suffice to say that the programming language that has been picked is different from the one initially proposed. With that being cleared out, let's have a look at the first part of the chapter, and that is the processing of the input image.

6.1 Processing of the Input Image

Any images are actually made up of pixels, the very basics of an element in an image. Therefore, any image processing task requires the system to be able to extract the properties / value of every pixel in order to manipulate / process each pixel. Using Microsoft Visual Basic 6.0, the image or every pixels of the image is being stored in a form of a dynamic array where the value for each pixel is stored to its (x, y) coordinate. The 2 main functions that are being used to extract and set the pixels are as shown below.

Getpixel()

Getpixel Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long) As Long

- This function retrieves the red, green and blue (RGB) color values of the pixel at the specified coordinate (x , y). It refers RGB to every pixel.

Setpixel()

Setpixel Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long, ByVal crColor As Long) As Long

- This function set the pixel at the specified coordinate to the specified color.

Both functions are widely used in / throughout in the program. Because of that both the functions above are being declared in the module for it to be easily accessible throughout the system.

6.2 Implementation of the Algorithm

As was proposed in chapter 4, the algorithm for recognizing faces consists of two main parts based on the technique of face preprocessing and face matching / recognition. Since the system will be built using Microsoft Visual Basic 6.0, the coding of both face preprocessing and the face recognition part is indeed a daunting task, and with that being said, the coding would be a little longer than that of MATLAB. Although the coding was longer, but the recognition time and preprocessing time was faster than if the system

were to be built using MATLAB. This is because MATLAB has a vast toolbox and while that is good and easier because we can call predefined functions from it, the time it takes to search through the toolbox will be a hindrance to the system in terms of recognizing time and preprocessing time. That was the main reason in using Microsoft Visual Basic 6.0 as the tool of choice. The coding for the face preprocessing and the face recognition part is as shown below.

6.2.1 Face Preprocessing

The face preprocessing part is the first part of the system that the user will come across before the user could do the recognition part. By default, when the preprocessing button is being triggered, 15 variation of face images shall be read and turn into eigenimages. Each of the eigenimages shall then be put in an array and eventually after the number of faces that was meant to be preprocessed had all being processed, the collective eigenimages of the various face images shall be displayed by the system. The coding of this part of the system is as shown below:

```
For i = 1 To NumF
    For j = Asc(start) To Asc(stops)
        NumFaces = NumFaces + 1
        ReDim Preserve FaceFileNames(NumFaces)
        ReDim Preserve Faces(xDiv, yDiv, NumFaces)
        ReDim Preserve EigenFaces(xDiv, yDiv, NumFaces)
        FaceFileNames(NumFaces) = App.Path & "\images\face" & i & Chr(j) & ".jpg"
        Picture1.Picture = LoadPicture(FaceFileNames(NumFaces))
```

```

For xBase = 0 To xDiv - 1
    For yBase = 0 To yDiv - 1
        xLow = (xBase * (Picture1.ScaleWidth / xDiv))
        xHigh = (xBase * (Picture1.ScaleWidth / xDiv)) + (Picture1.ScaleWidth / xDiv) - 1
        yLow = (yBase * (Picture1.ScaleHeight / yDiv))
        yHigh = (yBase * (Picture1.ScaleHeight / yDiv)) + (Picture1.ScaleHeight / yDiv) - 1
        CellSum = 0
        For xSub = xLow To xHigh
            For ySub = yLow To yHigh
                PColor = GetPixel(Picture1.hdc, xSub, ySub)
                RedLevel = PColor And 255
                GreenLevel = (PColor And 65280) / 256
                BlueLevel = (PColor And 16711680) / 65536
                GrayLevel = (RedLevel + GreenLevel + BlueLevel) / 3
                'GrayLevel = (PColor / MaxColor) * 255
                CellSum = CellSum + GrayLevel
                'SetPixel Picture2.hdc, xSub, ySub, RGB(GrayLevel, GrayLevel, GrayLevel)
            Next ySub
        Next xSub
        CellAvg = CellSum / ((Picture1.ScaleWidth / xDiv) * (Picture1.ScaleHeight / yDiv))
        Faces(xBase, yBase, NumFaces) = CellAvg
    Next yBase
Next xBase
Picture2.Refresh
DoEvents
Next j
Next i

```

6.2.2 Face Recognition / Matching

After the face preprocessing part has been done, the recognition part takes place. The recognition / matching part does as the name implies; match the input face image with that of the face images in the database based on the eigenimages that has been trained. The recognition coding is as shown below:

```
For xBase = 0 To xDiv - 1
    For yBase = 0 To yDiv - 1
        TestEigenFace(xBase, yBase) = TestFace(xBase, yBase) - FaceTemplate(xBase, yBase)
    Next yBase
Next xBase

MinEigenDiff = 99999999999#

For i = 1 To NumFaces
    TotalEigenDiff = 0
    For xBase = 0 To xDiv - 1
        For yBase = 0 To yDiv - 1
            TotalEigenDiff = TotalEigenDiff + Abs(TestEigenFace(xBase, yBase) - EigenFaces(xBase, yBase, i))
        Next yBase
    Next xBase
    If MinEigenDiff > TotalEigenDiff Then
        MinEigenDiff = TotalEigenDiff
        MinEigenIndex = i
        Debug.Print "Index: " & MinEigenIndex & " Diff: " & MinEigenDiff
    End If
Next i
```


When an input image is inserted, that image shall be trained and changed to a form of eigenimage. The eigenimage then shall be used to match with the face images in the database, and the face image when compared, will have a value for each comparison, in which in this system, that value is called MinEigenDiff, which means minimum eigen differences. The ones that has the lowest MinEigenDiff when compared with the input image shall then be known as the match to the input image, and thus shall be displayed as the matched face image.

6.3 Implementation of the System

The system works by first retrieving the eigenimages of the preprocessed images. For this to happen, the preprocessing button must be triggered first. The system then shall preprocess the images and store the eigenimages of the face images that has been trained in an array, which shall be displayed after preprocessing period has finished. The system then shall need a face image as an input into the system. After the system has an input image, the user is required to trigger the recognition button. This will lead to the input image being changed to an eigenimage, and stored memory. The eigenface of the input image is then match with those in the database, and the ones that are of the lowest MinEigenDiff's value of all the images in the database shall be displayed as the matched face image.

The other codes to the program are as shown in the previous segment. All the codes are done in the form that the action takes place. However, there are few global variables

that are being declared under a separate module file, such as the SetPixel() and the GetPixel(). The code to the module is as such:

```
Public Declare Function GetPixel Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long) As Long
```

```
Public Declare Function SetPixel Lib "gdi32" (ByVal hdc As Long, ByVal X As Long, ByVal Y As Long, ByVal crColor As Long) As Long
```

```
Global FaceTemplate() As Integer
```

```
Global Faces() As Integer
```

```
Global EigenFaces() As Integer
```

```
Global NumFaces As Integer
```

```
Global FaceFileNames() As String
```

```
Sub GetExample()
```

```
Dim PColor As Long
```

```
PColor = GetPixel(Picture1.hdc, X, Y)
```

```
End Sub
```

```
Sub SetExample()
```

```
SetPixel Picture1.hdc, X, Y, RGB(0, 0, 0)
```

```
Picture1.Refresh
```

```
End Sub
```


6.4 Discussion

The system was proposed using MATLAB, but because of certain unforeseen circumstances the system was actually built using Microsoft Visual Basic 6.0. This is something debatable indeed, but there are reasons behind the sudden change of the programming language of choice. Although there are many reasons regarding to the change in the programming language used, the main reason would be that of because Microsoft Visual Basic 6.0 is easier to understand and use, and furthermore it is also more convenient to do complex manipulation with the use of pointer based objects like that of the ones in Microsoft Visual Basic 6.0. From the system's point of view, the system that was built using MATLAB usually are a lot slower than that of the ones built using Microsoft Visual Basic 6.0. Therefore concludes the discussion regarding that of the sudden change of the programming language of choice.

6.5 Chapter Summary

This chapter explains in detail the algorithm for the face preprocessing and the face matching / recognition, which are both main methods in which would lead to the recognition of the face images. The program itself was built using Microsoft Visual Basic 6.0, which was a big change from what was initially proposed. The reasons for using Microsoft Visual Basic 6.0 have been discussed under segment 6.4, which is the discussion segment. The system was implemented successfully.

CHAPTER 7

TESTING

Testing is indeed one of the vital part when building softwares. In project such as these, we need to really know the bugs and anything that could really be a nuisance to the system. Thus we need to do certain number and types of testing in order to come up with a system that is bugs-free, or nearly bugs-free. There are three types of testing that was opted for this project, namely the unit testing, integration testing and system testing.

7.1 Unit Testing

Unit testing is a white box testing technique in which the single main reason for it would be that of to point out 2 types of errors that are quite common when building software; which are the algorithmic errors and the computational errors. All in all, the most basic and common of errors done are listed below, in regards to this project:

- If – Else error, normally resulting in the system's logical decision making goes haywire, which is indeed grave if it were to go unattended.
- Looping errors, such as For loops and While loops, which will have a major impact upon the control flow of the system
- Coding Errors, which are really just normal coding errors, resulting from lack of focus. Coding errors are hard to find because they are hidden and sometimes our

lack of understanding regarding the programming language will be a big setback in finding the errors and debugging them.

Basically, those are the three types of errors that were found in most systems in their testing phase, and it is also the types of errors that were found when building the system for this project. Thus, careful steps were taken in order to combat these errors that lurk within the system while in its testing phase.

- The codes are being checked again and again to understand the flow of the codes and also to understand the logical flow of the codes. The errors that occur every time the system is being tested shall be corrected. After the system has been corrected of its previous errors, and after new lines of coding or modules have been added, the system is then tested again. This is more like an iterative way to combat those errors.
- The flow and looping of the system are being studied again and again, to better understand the way of its flow, and if the previous flow of the system is full of faults, then some minor modifications shall be done to the codes to counter with the error.
- The system is then tested under many circumstances to see how the system works in various ways. This was done as to let the errors that weren't really obvious before this surface and by doing so we can change and heal that part of the system in regards to any coding or logical errors.

7.1 Integration Testing

Integration testing is also a white box testing technique. The only difference is that the unit testing is more towards that of coding errors, while integration testing is, as the name implies, more of a technique to test the interface between the functions and its subroutines. Sometimes these errors do not show while in the unit testing phase, but then it arises when we do integration testing. There are many methods of doing integration testing, but the methods that was adopted for this project is as such:

- Randomly test each function to its subroutines, calls its subroutines and when an error occur, note that error and fix it.
- Do a bottom up approach, which is to say, when a function A calls function / subroutine B and function / subroutine B calls that of the function / subroutine C, thus the testing is being done on C first, and walk its way up to A. If there are any errors, it will surely show in the bottom up process.

Both the unit testing and integration testing are being done concurrently, as it is impossible to separate each other, for both these techniques depend on each other.

7.3 System Testing

System testing is important as it tests the function and the performance of the system. This is the benchmark for the system's capabilities and abilities. Below are the function testing and performance testing explained.

7.3.1 Function Testing

System testing deals with that of the functional and non-functional side of the system. It simply tests if the system had fulfilled the functional and non-functional requirements correctly. The use case specified in Chapter 3 was referred to doing the function testing so that no use cases would be left out from the system.

7.3.2 Performance Testing

The system's main objective is to take an input image and match it with the one in the database. If there is a match or something similar in the components of the images, then the system will output the image as the image matched. Although the system may look simple on the surface, it is indeed contrary to what lies beneath it. The system's method of recognizing an image depends on the Eigen Differences (MinEigenDiff) between the input image and the array of trained images. When compared, the image with the lowest MinEigenDiff shall be the image of choice and shall be known as the matched image. The PCA Face Recognition System has a division bar, which is used to set the picture's

pixel division to a certain value. The higher the division bar is being raised, the calculation for the images becomes a bit more complicated and thus, recognition response time gets a bit longer.

The purpose of the performance testing is to measure the accuracy of the Face Recognition System in recognizing faces. Although it is not 100% accurate, but the recognition rates gets better with each trained images. Image must be 24 bit grayscale face image with the width and height of approximately 80 and 80 respectively in pixel unit. Although the system has been trained using grayscale images, a standard RGB color images could also be used (an updated research and testing upon the system shows that the system can be used to recognize color images).

The system is being put through a couple of tests, to test the recognition rate, the MinEigenDiff's value as the division bar is raised, the recognition response time as the division bar is raised, and also, from the scale of 1-10, when does the recognition rate gets to a point that the recognition could be considered as acceptable.

The Recognition Rate

The system consists of 15 individual's picture, with roughly about 3-4 variations of poses for each individual. If a person number 1 was to have 3 variations of poses, then the name of his \ her faces would be face1a, face1b and face1c. With that being said, let us have a look at what the outcome would be if those 15 face images were to be trained from variation a-c. The system was tested using 5 face images.

Division\Faces	Face14a.jpg	Face15b.jpg	Face1c.jpg	Face2a.jpg	Face3b.jpg
0	Cannot Recognize!				
10	Yes	Yes	Yes	Yes	Yes
20	Yes	Yes	Yes	Yes	Yes
30	Yes	Yes	Yes	Yes	Yes
40	Yes	Yes	Yes	Yes	Yes
50	Yes	Yes	Yes	Yes	Yes
60	Yes	Yes	Yes	Yes	Yes
70	Yes	Yes	Yes	Yes	Yes
80	Yes	Yes	Yes	Yes	Yes
90	Yes	Yes	Yes	Yes	Yes
100	Yes	Yes	Yes	Yes	Yes

Table 7.1: Recognition Rates for various combinations of images and division values.

As can be seen the recognition rate is very good indeed under these circumstances. Although the table shows that the recognition rate is 100%, there are minor conditions where a face image could be wrongly recognized. These happenings might result from the same lighting condition or the same hair, nose, mouth, as the next person's image. Another possibility of this state is that if the variation is minimized in training, such as instead of training variations a-c, we just train a-a, which will result in the face images being recognized wrongly.

Division Value versus MinEigenDiff's Value

This test strictly shows the changes in value for the MinEigenDiff's value as the Division Bar is raised. As for the face images, we opted to train from variations a-b, and purposely feed face images of the same face image but of the variation c to see the differences in MinEigenDiff's value as the division bar is raised.

Division\Faces	Face14c.jpg	Face15c.jpg	Face1c.jpg	Face2c.jpg	Face3c.jpg
0	Cannot Recognize!				
10	597	283	745	401	603
20	1325	4024	2209	2519	3902
30	3248	9940	5986	6116	9510
40	6553	18853	11268	11291	17774
50	10795	30211	18397	18009	28354
60	15440	43593	28556	26373	41358
70	22254	62671	40809	38050	59410
80	30041	82798	54713	50377	78615
90	37997	104809	74776	66708	101714
100	48157	129105	94518	84054	126986

Table 7.2: MinEigenDiff values for various combinations of images and division values.

As can be seen, the values get bigger as the division bar is raised. This was expected as the division bar is raised, the pixel value for X-axis and Y-axis is also raised, thus resulting in higher MinEigenDiff value.

The Recognition Response Time as the Division Bar is Raised

From the previous section we can see that the value of MinEigenDiff gets larger as the division bar is raised. In this test however, we tested out the response time of the system as the division bar is raised.

Division	Time (In Seconds)
10	0.167
20	0.173
30	0.186
40	0.200
50	0.217
60	0.249
70	0.280
80	0.308
90	0.361
100	0.382

Table 7.3: Divisions and response time

Not surprisingly, the response time gets slower as the division bar is raised. This is because as the division bar is raised, it regulates how the pixel of the picture should be. Saying that if the division bar is raised up to 50, then the division for X-axis and Y-axis for the images is 2500 pixel. If the division bar is raised to 80, then the pixel of the picture shall be roughly 6400 pixels. Of course these numbers are just estimation, but it show how the by changing the division bar's value, the more computational power it takes to process an image.

Division range of 1-10 and number of right matches made by the system

Another study was conducted in terms of when the systems really make a good recognition rate. The meaning of the previous statement is that, from a division range of 0-10, when the system really matches the input images with the right images (image of the same person). The table bellows shows the result of the study:

Division \ Faces	Face14c.jpg	Face14c.jpg	Face14c.jpg	Face14c.jpg	Face14c.jpg	Right Matches
1	No	No	Yes	No	No	1/5
2	No	Yes	Yes	No	No	2/5
3	Yes	No	Yes	No	No	2/5
4	Yes	Yes	Yes	Yes	No	4/5
5	Yes	Yes	Yes	No	No	3/5
6	Yes	Yes	Yes	Yes	No	4/5
7	Yes	Yes	Yes	Yes	Yes	5/5
8	Yes	Yes	Yes	Yes	Yes	5/5
9	Yes	Yes	Yes	Yes	Yes	5/5
10	Yes	Yes	Yes	Yes	Yes	5/5

Table 7.4: Division range of 1-10 and number of right matches made by the system

As can be seen the recognition rate was quite bad at the start, and is shows that the smaller the pixel divisions are, the harder it is for the system to get a good recognition rate of the images. Also, from the table above, we can deduce that the system's recognition rate, from the division range of 1-10, takes a good turn when it reached the 7th division, and what follows afterwards shows that the recognition becomes better with every level of division raised.

7.4 Discussion

As was seen above, the performance test shows that the recognition rate was really good, where the recognition was 100%. But of course there are conditions where recognition might go wrong, perhaps in different lighting condition or in minimizing the number of variations to be preprocessed. It was also shown that the MinEigenDiff's value and the response time value gets larger as the division bar is raised, and that the recognition value is gets to a level that is acceptable to be used for recognizing faces when it reaches the 7th division.

Earlier in the chapter we showed the method for doing unit testing and also integration testing, which was both carried out concurrently as they depend on each other. The function test showed that the system must meet the functional and non-functional requirements. One thing that was told in this chapter is that instead of using grayscale images, we can also use standard RGB color images. This is indeed something good because if we were to have color images we wouldn't want to change them into grayscale before we get to use the system. Instead, it would be better if we could just feed it into the system and the system changes the images into grayscale. Indeed the system could do that, for some minor modification was done to the code as to cater for color images. the expectancies would be that the system would fair the same whether the system uses grayscale of color images, because both will be changed to grayscale and then the recognition part would take place. Thus it is not a problem for this system to identify and recognize color images.

7.5 Summary

CHAPTER 8

SYSTEM EVALUATION

The PCA Face Recognition System's performance is based mainly in its ability to do accurate recognition of face images. As was seen, the face recognition rate was 100%, and other test was also done regarding how the MinEigenDiff and response time value changes as the division bar is raised. The unit testing method and integration method was shown earlier in the chapter, along with the system's testing method.

After all the phases and testing has been done, now we want to see how the system itself as a whole. The following is an example of how the system works.



Figure 3.1 Recognition of face

There are four boxes in the picture above, and each one of them represents a different type of image. As for that, here are the labels to the four boxes above.

CHAPTER 8

SYSTEM EVALUATION

&

CONCLUSION

8.1 Results

After all the phases and testing has been done, now we shall have a look at the system itself as a whole. The following is an example of how the program works.

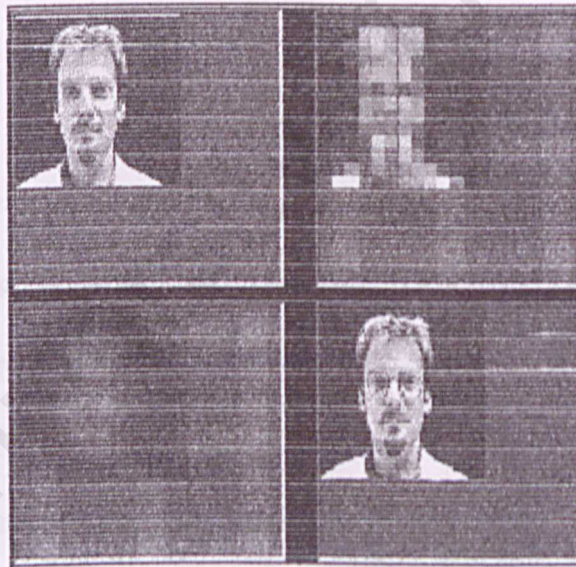


Figure 8.1: Recognition of Face

There are four boxes in the picture above, and each one of them represents a different type of image. As for thus, here are the details to the four boxes above.

- **Top Left:** Is the input image, and it is the image that will be used for the system to find the face image in the database that matches it.
- **Bottom Left:** It is the eigenimages of the face images that has been trained before the input picture is being fed. After the process of preprocessing the face images that has been trained are all put in an array as eigenimages, and after the preprocessing stages, the system will display the eigenimages.
- **Top Right:** Is the input image eigenimage, after it has been through a process of processing before it is being matched to the array of eigenimages.
- **Bottom Right:** The matched image in the database. As can be seen, the image that has been matched with the input image is of a different variation (face1c, face11, etc), but because they share the same principal components, they are matched as the same person.

The image above shows that the person is one and the same. If one were to look carefully, the input image is the face of the person without his spectacles, but the matched image is the person's face with his spectacles on. Thus this concludes that even though the person is wearing something else on his face, given that the principal components of the person is visible and can be calculated, the person can be recognized by the system.

8.2 Strength & Weaknesses

Like any systems of any kind, we cannot run away from the fact that there are such things as strengths and weaknesses. Below are listed the system's strengths and weaknesses.

Strengths

- The interface is easy to use and user friendly.
- The system requires no prior knowledge regarding the topic. All the user needs to do is feed in an image and click away at the recognition button, and the system will do the rest.
- The designs of the system is rather straightforward, therefore users will easily get used to it.
- The system shows the eigenimages, the input image, the input image's eigenimage and also the matched image, all in one interface. The user can have a look at a variety of versions of the image at once.

Weaknesses

- Although the system has been known to have 100% accuracy in recognizing face images from input image, there are conditions where the system might be fooled by the image itself, and therefore, we cant really say that it will cater for all types of images, for there are factors such as background colors, variation of pose and others that might hinder the system to recognize the person's image.
- The system only caters for images with that of the face of the person facing straight to the camera. It does not cater for images from the sides, back, top and such. Images of such poses might not be recognized by the system, and it would not be able to help the system in recognizing face images.

As to date, there are no Face Recognition systems that can produce strictly 100% accuracy.

8.3 Future Application

Understanding PCA Face Recognition is a good way start into face recognition, as the field is quite large on its own. In the future, these applications might be incorporated into security systems without causing any flaws or such. Also, face recognition can be used together with other biometric recognition process, such as voice and iris recognition. Face recognition works best if it were to be incorporated with other biometric recognition process, and saying that if it were to be coupled with face detection algorithm in the same system as the face recognition system, the system itself

would be a good security system indeed. The problem now is just the question of how much trust can be given to these kind of systems, and can they produce the result that are acceptable to the community at large? Time will tell, and as for now, the search for the most reliable and usable face recognition system goes on.

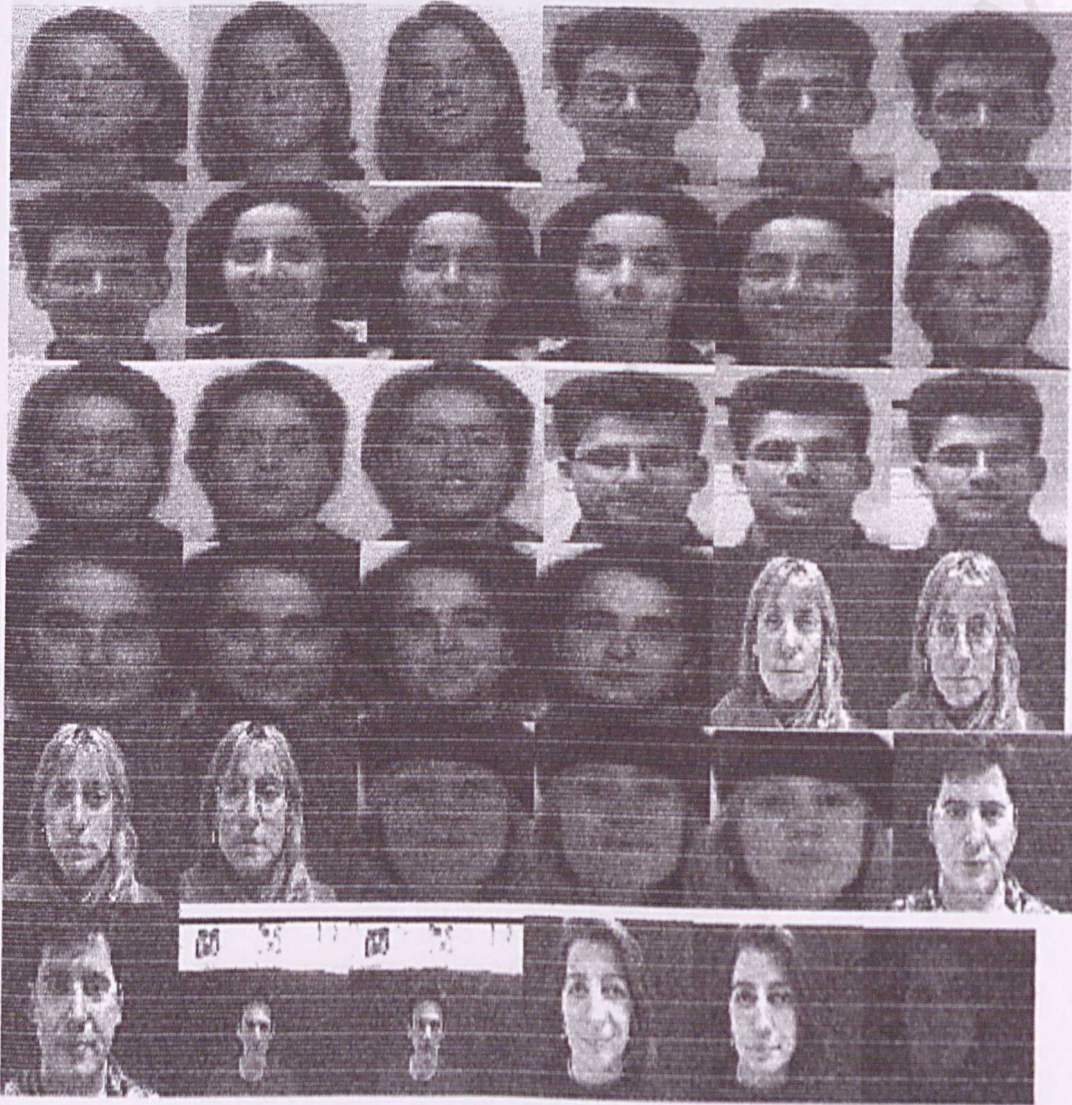
8.4 Summary

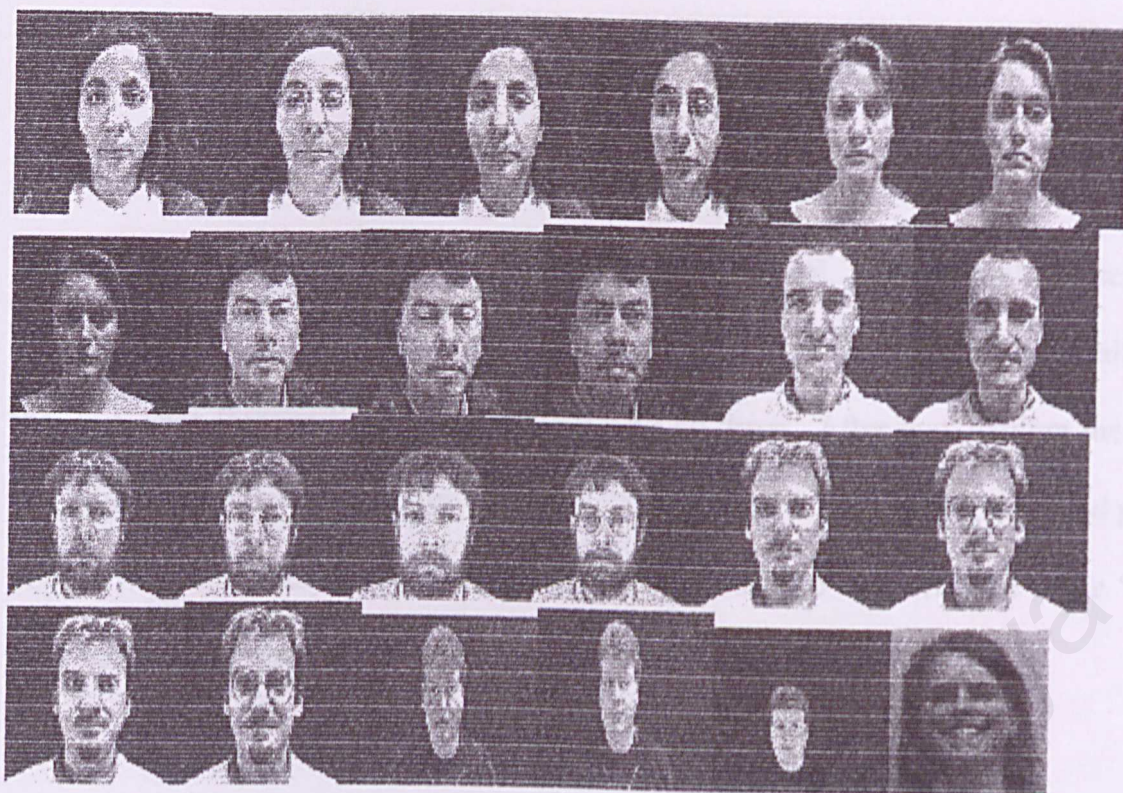
The PCA Face Recognition system was successfully implemented. Although the system sports a 100% recognition rate, but there are conditions that might hinder that fact, for there are no systems that are that perfect. Like that of other systems, it comes with strengths and weaknesses, and is shown in this chapter. The system was build using Microsoft Visual Basic 6.0, and due to the complexities of its processes, the time it took to learn the algorithm, to learn how to code it, the basics of it all, and that of image manipulation, took quite a big chunk of the time to build the system. Gladly, to come to a conclusion that it was all worthwhile, and as for the future of face recognition, the journey is still far ahead. But rest assured, we are catching up with it.

APPENDIX

APPENDIX A. FACE DATABASE

The face database consists of 54 face images of 15 people altogether. The face database is obtained from the Sluggish Software website [10], which was used for their face recognition system, and the face database consist also of 9 females and 6 males, each are about 20-35 years of age.





APPENDIX B: USER MANUAL

PCA FACE RECOGNITION SYSTEM

Welcome to the PCA Face Recognition system user manual. This manual will serve as a way to educate users into having the know-how of using the system. First of all, insert the PCA Face Recognition system cd into your cd drive. After that, explore the cd and you will see a folder that goes by the name "SiapThesis!!". Copy that folder and put it in your C:\ drive. After that, search the folder, and click the icon with the name "PCA". That will start off the system.

The next section shall discuss the variety of sections of the PCA Face Recognition system and how to use it.

Interface

Below are shown the 5 main sections of the PCA Face Recognition system.

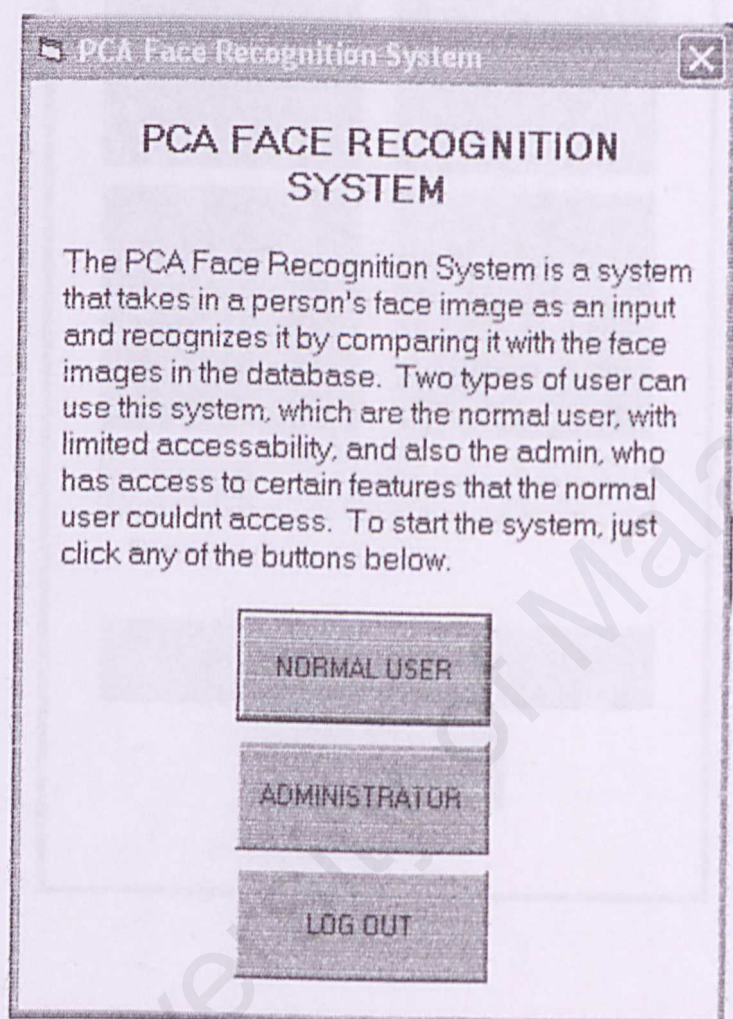


Figure 1: PCA Face Recognition System

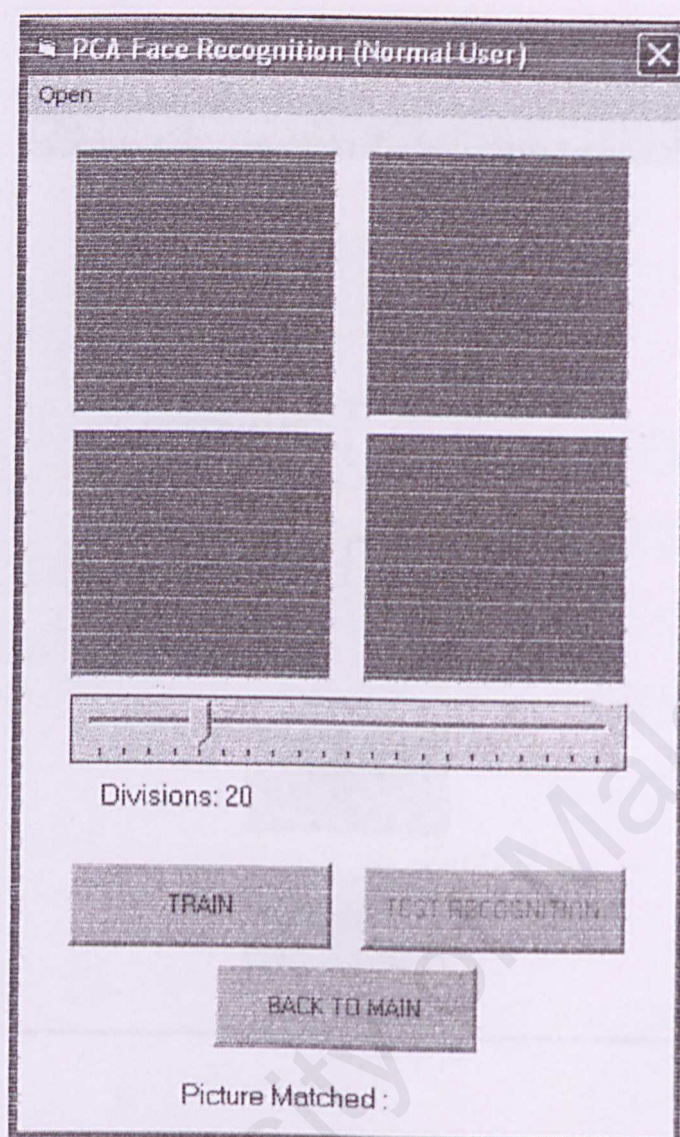



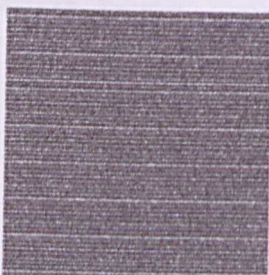
Figure 2: Normal User's Section

A screenshot of a Windows-style dialog box titled "Login As Administrator". The dialog box has a standard title bar with a close button (X) in the top right corner. Inside the dialog, there are two text input fields. The first field is labeled "USERNAME" and the second is labeled "PASSWORD". Below the input fields, there are two buttons: "ENTER" and "BACK TO MAIN". The dialog box is centered on the screen. A large, faint watermark "University of Malaya" is visible across the background of the page.

Figure 3: Login as Administrator

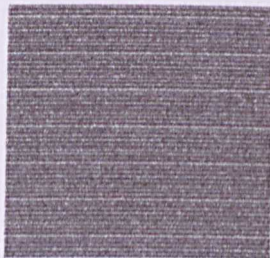
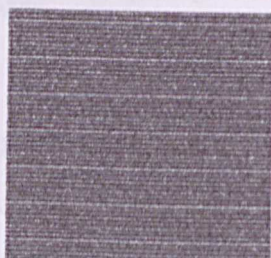
PCA Face Recognition System (Administrator) [X]

Open

Num. Of Pictures In Database :

UPDATE

Enter Alphabets Ranging From a To c:

To

UPDATE

Recognition Time (in seconds):

Divisions: 20

TRAIN

TEST RECOGNITION

CREATE NEW PASSWORD

Path And Picture Matched :

LOG OUT TO MAIN

Figure 4: Administrator's Section

Figure 5 shows a 'Create New Password' window. It features a title bar with a close button. The main area contains two text input fields. The first field, labeled 'USERNAME', has the text 'admin' entered. The second field, labeled 'PASSWORD', has six 'x' characters entered. Below these fields are two buttons: 'SAVE' and 'BACK TO MAIN'.

Figure 5: Create New Password (default username and password: admin)

Those were the five sections in the PCA Face Recognition System. Although there are five interfaces, the ones that are of utmost important to us is the face preprocessing and recognition section, and there are two versions of it, and that is the normal user version and also the administrator version. Both are the same albeit slightly different in terms of the number of functions. Therefore, as for an example of how to use the system, the administrator version of the system shall be used instead.

EXAMPLE: ADMINISTRATOR FACE RECOGNITION SECTION.

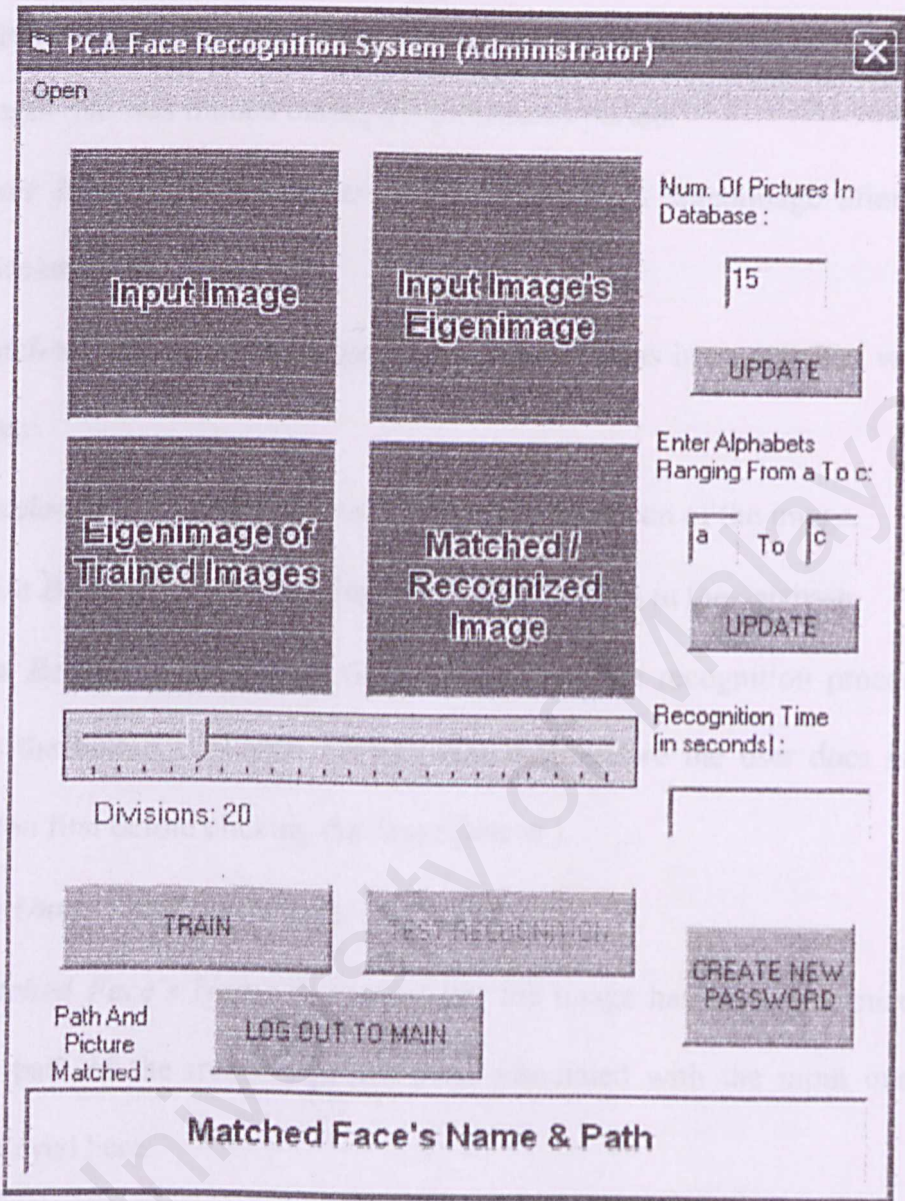


Figure 6: Administrator Face Recognition section

As can be seen, there are more features in this section than that of the normal user's section. However, the recognition process is quite the same. A bit of an introduction will ease the use of the system.

- **Open Menu:** To open and browse to pick face images.
- **Input Image:** The place where the input image is being displayed and captured
- **Eigenimage of Trained Images:** Displays the Array of Eigenimages of the face images that was trained during the preprocess period.
- **Input Image's Eigenimage:** The Input Image's eigenimage after it is being processed
- **Matched \ Recognized Image:** The image that has been matched with the input image.
- **Division Bar:** Change the X-axis and Y-axis division of the image.
- **Train Button:** To start the preprocessing of images in the database.
- **Test Recognition (Slightly Grayed):** To start the recognition processed (notice that the button is slightly grayed; this is to ensure the user does not click the button first before clicking the Train Button.).
- **Log Out to Main:** Back to the main page.
- **Matched Face's Name & Path:** After the image has been recognize, the name and path of the image that has been associated with the input image will be displayed here.
- **Num. of Pictures in Database:** To change the number of images that will be trained by the system.
- **Enter Alphabets From a to c:** Since there are many variations of the person's image (face1a, face1b, etc), thus this part regulates how many variations that the user are able to train. Within the range of a-c, the user can change by either entering variations a-b (2 variations) or a-a (1 variation).

- **Recognition Time:** The time that it takes for the system to do recognition of a single image.
- **Create New Password:** To create new password for the administrator section.

With that entire aside, let us have a look at how the system works in recognizing face images. After the user has entered the Administrator or Normal user section of the face recognition system, the interface shall be like the one shown in Figure 2 and Figure 4. For example purposes, we shall use the Administrator section of the face recognition system.

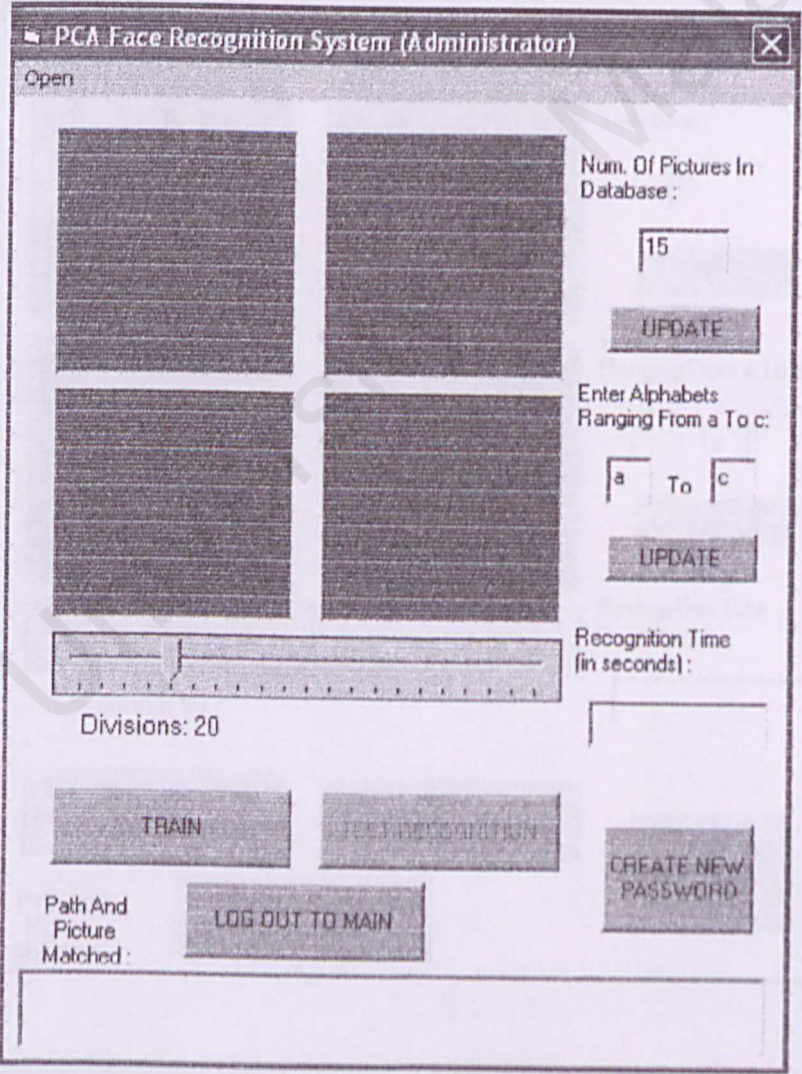


Figure 7: Administrator Face Recognition section

The first thing for the user to do is to click the Train Button. But before that, the user can actually determine the division of X-axis and Y-axis of the face images to be preprocessed. In Figure 7, we can see that by default the division bar is set at 20. In Figure 8, we can see that the division bar has been set to 50 and the Train Button has been clicked.

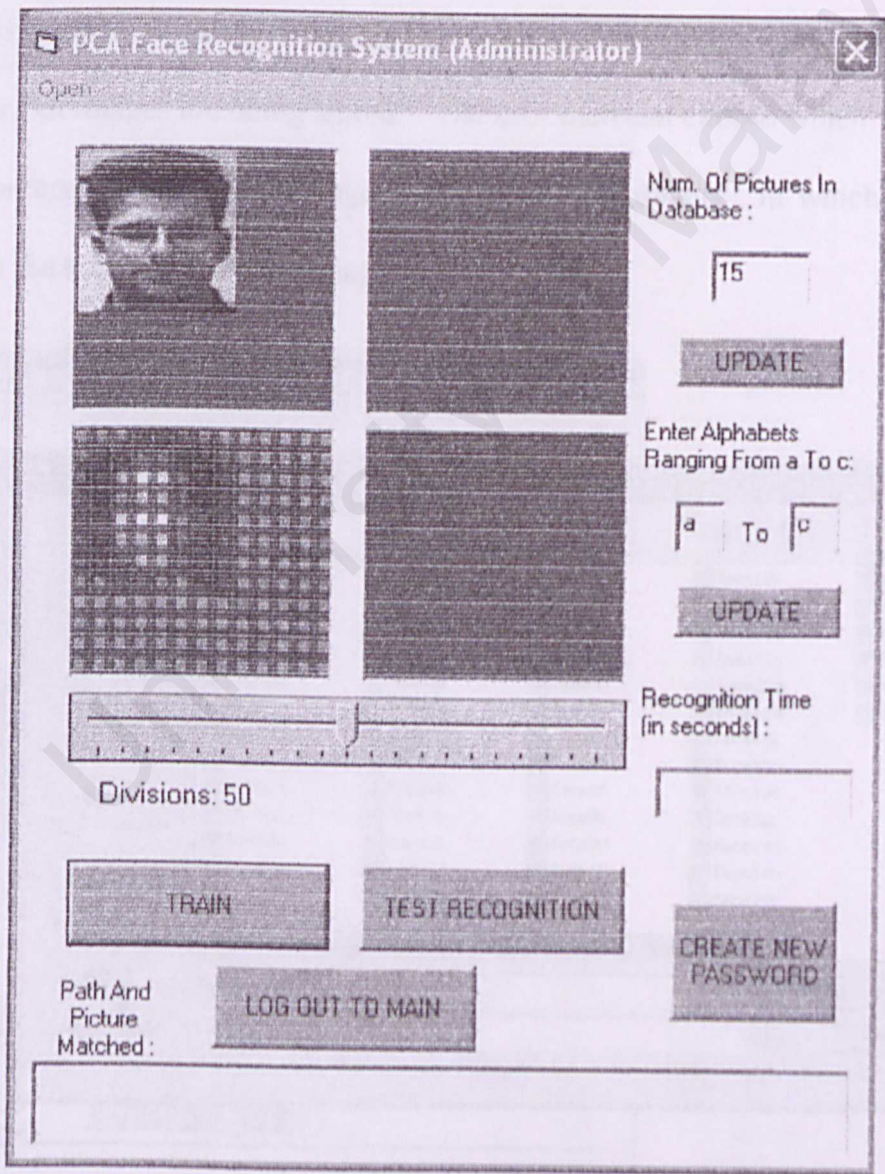


Figure 8: Administrator Face Recognition section (Train Button Initiated)

During the preprocessed session, we can also see that in the input image's picture box is being used to show the images that have been used in its preprocessing session. We can also see that the default number of images that has been trained is 15, with the variations of a to c.

After the preprocessed session, the user can click the Open Menu at the top left hand corner of the system. After clicking the Open Menu, it will open up the folder in which the databases of images are being stored. The user then can choose which images to use (for convenience, the user can change the view to “thumbnails”, in which the user can actually see the thumbnails of the images).

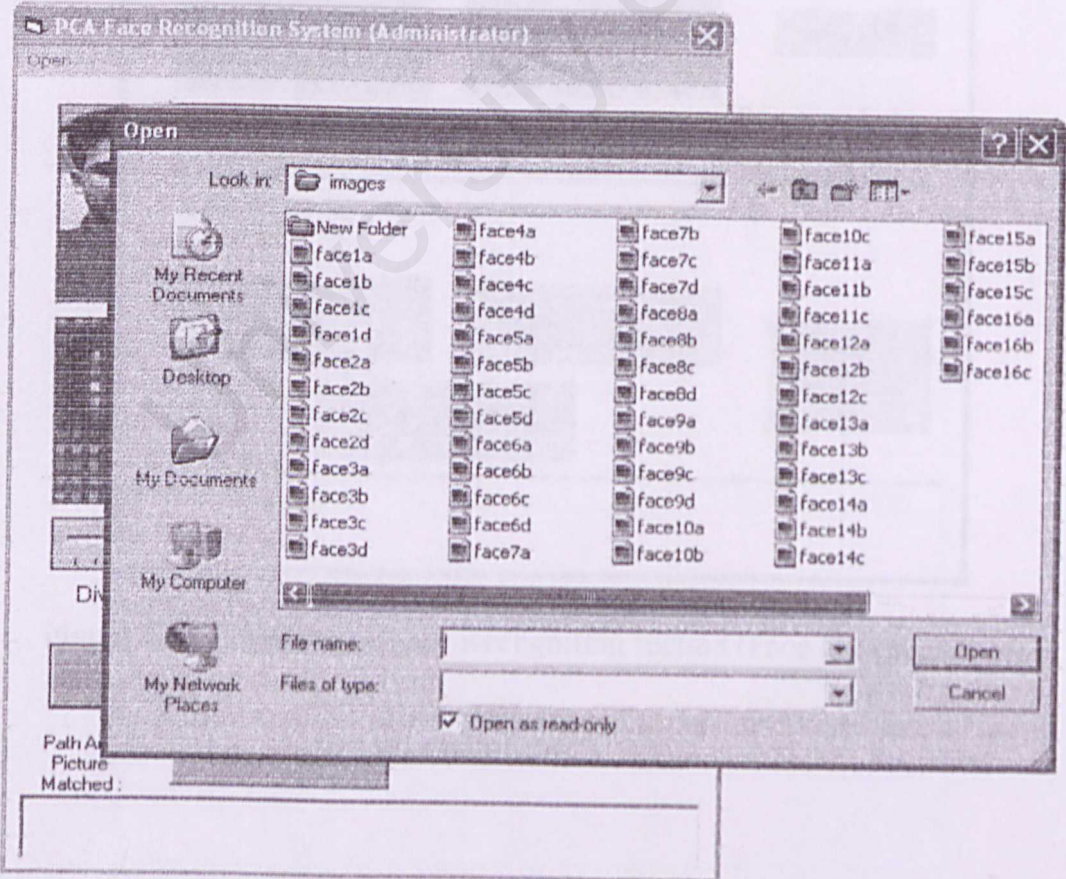


Figure 9: Administrator Face Recognition section (Open Menu Initiated)

After the user has chosen the face image to be used, the face image will be displayed in the input image's picture box. For this example, the face image that used was that of the face1c.bmp.

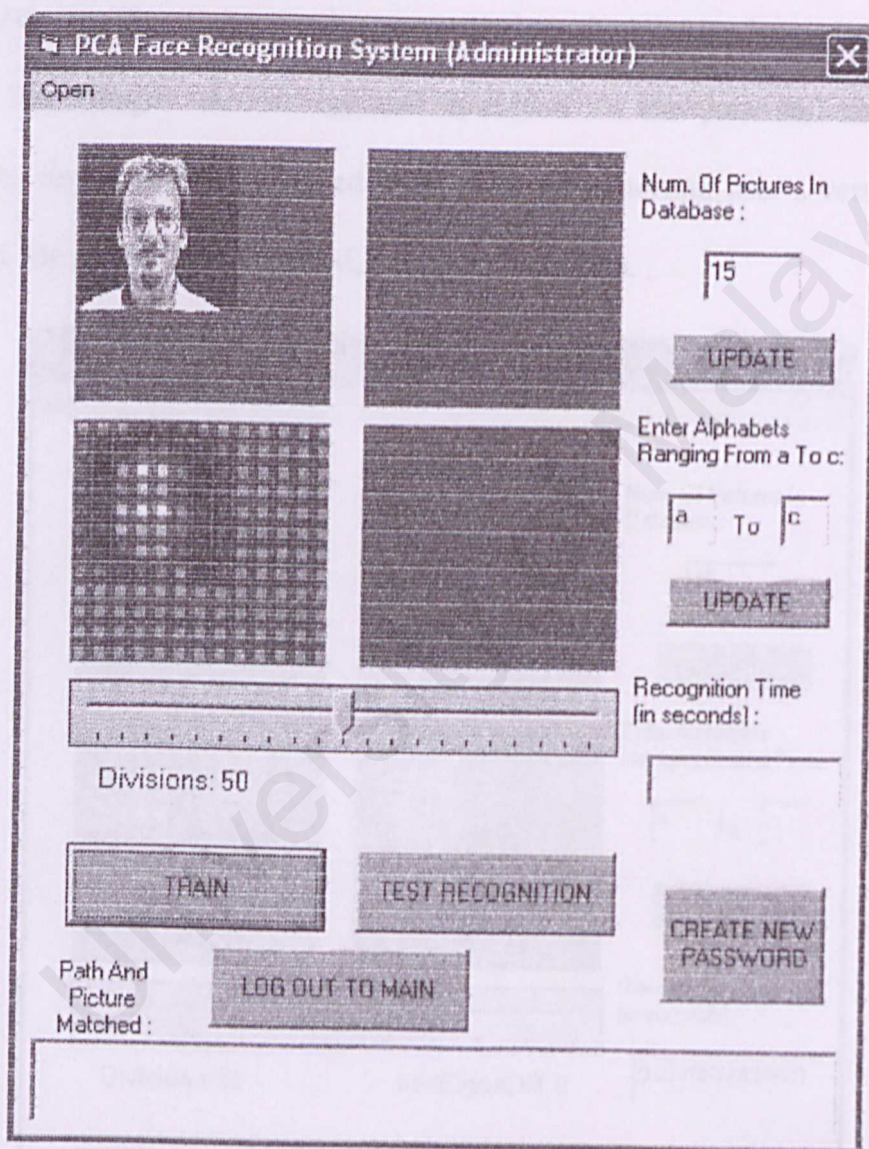


Figure 10: Administrator Face Recognition section (Face Image Chosen)

After that, all that is left is to click the Test Recognition Button in order for the system to do recognition. The first thing the system will do is change the input image into an eigenimage. Directly after that, the system will then matched the input image's eigenimage with that of the array of eigenimages of the images that has been preprocessed in Figure 8. The matched face image will then be displayed, and the value of the MinEigenDiff will be displayed just below the picture box of the matched / recognized face image. As can be seen in Figure 11, the path and filename of the matched face image is also displayed (note that in the normal user's version, only the name of that file matched is displayed, and the path is not).

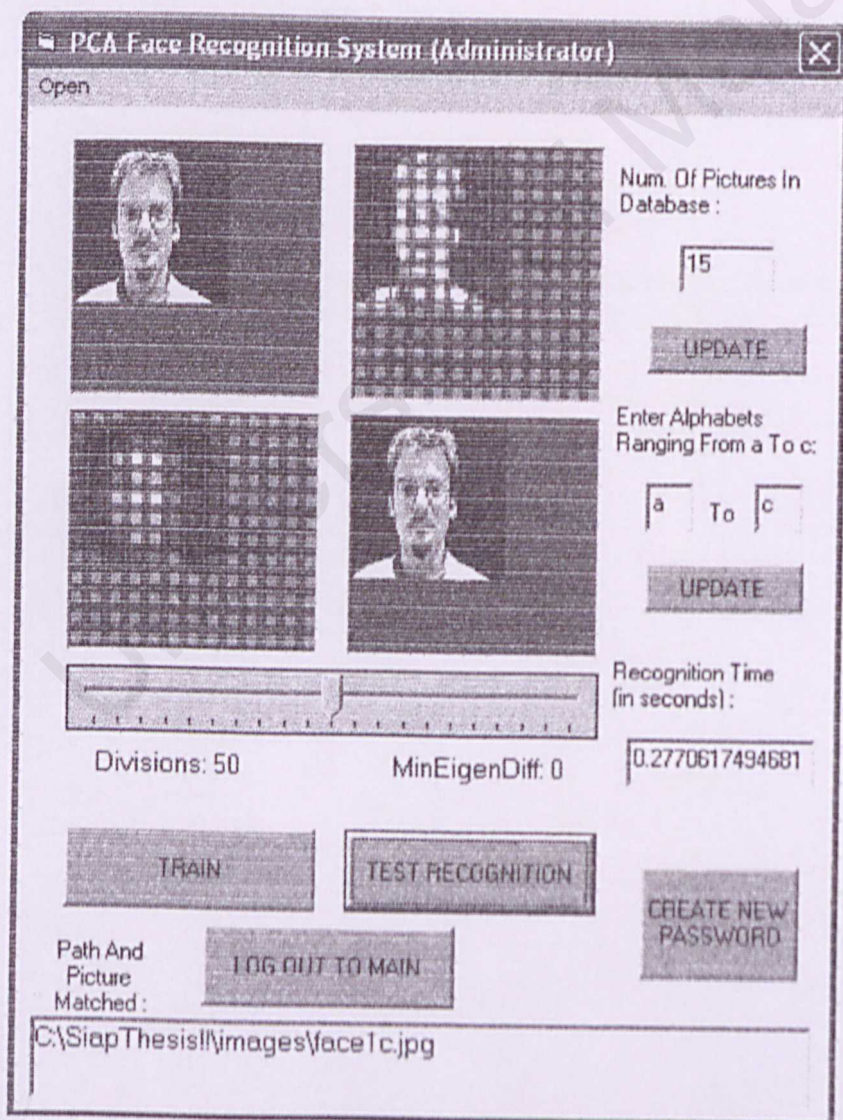


Figure 11: Administrator Face Recognition section (Test Recognition Button Initiated)

Other things to be noted are that the recognition time is being displayed in its text box and that the MinEigenDiff are also being displayed just below the matched / recognized image picture box.

-The End-

REFERENCES

- [1] "Principal Component Analysis and Neural Network Based Face Recognition "(Qing Jiang, 1998)
- [2] "Face Recognition Using Eigenfaces" (Matthew A. Turk and Alex P. Pentland, MIT, 1991)
- [3] "Computer Recognition of Human Faces" (Kanade, 1977)
- [4] "Elastic Graph Bunch Mapping" (Wiskott *et al*, 13th May 2003)
- [5] "Recognition Using Class Linear Projection" (P. Belhumeur, J. Hespanha, and D. Kriegman, Yale University, July 1997)
- [6] "The definition for Software Development Process"
http://www.wordiq.com/definition/Software_development_process
- [7] The Rational Unified Process
<http://www.menloinstitute.com/freestuff/whitepapers/rup.htm>
- [8] De Monfort University, "Object-Oriented Systems Design the Unified Process"
www.cse.dmu.ac.uk/Modules/INFO/INFO2004/lecture_files/2004B03.ppt
- [9] "The UML Resource Page"
<http://www.uml.org/>
- [10] "Sluggish Software"
<http://www.fuzzgun.btinternet.co.uk/Downloads.htm>